The Dissertation Committee for Suvrit Sra
certifies that this is the approved version of the following dissertation:

Matrix Nearness Problems in Data Mining

Committee:

Inderjit S. Dhillon, Supervisor

Todd Arbogast

Joydeep Ghosh

Raymond Mooney

Vijaya Ramachandran

Matrix Nearness Problems in Data Mining

by

Suvrit Sra, B.E. (Hons.), M.S.C.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2007

Dedicated to my parents...
Ram D. Sra
and
Ranjit S. Sra
I owe you everything.

Acknowledgments

No enterprise, howsoever modest, is ever accomplished single-handedly. Putting behind the myth of the monolithic personality, it is thus in order, that I acknowledge gratefully the support provided to me my numerous people as I have journeyed through graduate school.

First, I would like to thank my adviser Inderjit Dhillon for providing me the opportunity to perform the research constituting this thesis in a most independent and unfettered fashion. I would also like to thank the members of my dissertation committee, namely, professors Todd Arbogast, Joydeep Ghosh, Raymond Mooney and Vijaya Ramachandran, for their support and advice.

A most important note of thanks goes to my sister, Misha Sra, for providing unwavering support throughout the duration of my Ph.D., not only urging me to finish, but also rendering help in various forms that made my life as a Ph.D. scholar much easier–Thanks! I am also deeply thankful to my parents for their love, encouragement and constant support, without which I would be nowhere. No meager words, such as the ones on this page, can express emphatically enough how grateful I am to both of them.

Numerous other people have supported me over the years; their support assuming multifarious forms such as: extensive discussions—both academic and my favorite, the alternate academic!, shared lamenting sessions (aka b**

sessions in less erudite terminology), encouragement, games, and doses of reality, among others. I have enlisted the names of these people below, and I apologize in advance, if I have missed a name due to blatant oversight or amnesia. In alphabetical order: Arindam Banerjee, Paolo Bientinesi, Hyuk Cho, Jason Davis, Robert van de Geijn, Yuqiang Guan, Prateek Jain, Stefanie Jegelka, Dongmin Kim, Brian Kulis, David Montoya, Elizabeth Pommier, Vinay Siddavanahalli, Matyás Sustik, Mitul Tiwari, and Joel Tropp. I thank you guys for the countless hours of discussions and other temporally or monetarily challenging pursuits that we have enjoyed (or perhaps not!) together.

I would also like to take this opportunity to send a 'shout-out' to my friends back home in India—your friendship and the memorable times that we've had together, have made my life more meaningful. I list some particular names here: Aman Kapur, Kush Kapur, Sanjeev (Bhaskar) Sharma, Ajay Sharma, Pawan Thakur, and Sandeep Vaidya.

Finally, I acknowledge the shaping role some of my teachers have had over the years, which was critical in catapulting me into Computer Science and thereby, into the web of research (which is not as exalted an escape from reality as many might presuppose it to be!).

Matrix Nearness Problems in Data Mining

Publication No. _____

Suvrit Sra, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Inderjit S. Dhillon

This thesis addresses some fundamental problems in data mining and machine learning that may be cast as matrix nearness problems. Some examples of well-known nearness problems are: low-rank approximations, sparse approximations, clustering, co-clustering, kernel learning, and independent components analysis. In this thesis we study two types of matrix nearness problems. In the first type, we compute a low-rank matrix approximation to a given input matrix, thereby representing it more efficiently and hopefully discovering the latent structure within the input data. In the second kind of nearness problem we seek to either learn a parameterized model of/from the input data, or the data represents noisy measurements of some underlying objects and we wish to recover the original measurements. Both types of problems can be naturally approached by computing an output model/matrix that is "near" the input.

The specific nearness problems that we study in this thesis include: i) nonnegative matrix approximation (NNMA), ii) incremental low-rank matrix approximations, iii) general low-rank matrix approximations via convex optimization, iv) learning a parametric mixture model for data, specifically for directional data, and v) metric nearness.

NNMA is a recent powerful matrix decomposition technique that approximates a nonnegative input matrix by a low-rank approximation composed of nonnegative factors. It has found wide applicability across a broad spectrum fields, ranging from problems in text analysis, image processing, and gene microarray analysis, to music transcription. We develop several new generalizations to the NNMA problem and derive efficient iterative algorithms for computing the associated approximation. Furthermore, we also provide efficient software which implements many of the derived algorithms.

With growing input matrix sizes, sometimes low-rank approximation techniques themselves can become computationally expensive. For such situations, and to aid model selection (the rank of the approximation), we develop incremental versions of low-rank matrix approximations, where the approximation is obtained one rank at a time. There are several applications of such a scheme, for example, topic discovery from a collection of documents.

We also develop methods based on large-scale convex optimization for computing low-rank approximations to the input data. Our approach can deal with large scale data, while permitting incorporation of constraints more general than nonnegativity if desired. Our approach has some beneficial

byproducts—it yields new methods for solving the nonnegative least squares problem, as well as ℓ_1-norm regression.

The next nearness problem that we look at is that of learning a parametric probabilistic mixture model for the data. Here one estimates a parameter matrix given the input data, where the estimation process is implicitly regularized to avoid over-fitting. In particular we solve the parameter estimation problem for two fundamental high-dimensional directional distributions, namely the von Mises-Fisher and Watson distributions. Parameter estimation for these distributions is highly non-trivial and we present efficient methods for it.

The final nearness problem that we study is a more typical matrix nearness problem, which is called *metric nearness*. The goal here is to find a distance matrix (i.e., a matrix whose entries satisfy the triangle inequality) that is "nearest" to an input matrix of dissimilarity values.

For most of the algorithms that we develop in this thesis, we also provide software that implements them. This software will be of use to both researchers and practitioners who want to experiment with our algorithms.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Truth is much too complicated to allow anything but approximations."
– John von Neumann

Machine learning (or data mining[1]) can be described as the discipline of automatically "learning" concepts from data either with, or without human guidance. The goal of most machine learning applications is to discover patterns within data so that certain properties (e.g., membership to a certain category) of hitherto unseen data can be predicted reliably. Naturally, representing data in an efficient and amenable manner is fundamental to most machine learning applications, be it for pattern classification, denoising, or for visualization. The importance of representation, especially for data analysis, cannot be overstated, for example consider speech recognition; to a human listener an audible sound makes much more sense than a list of numbers, though for computation the latter list might be more preferable. Deeply intertwined with the task of representing data is the process of modeling it using either probabilistic models or other mathematical structures.

[1]Usually people associate data mining with unsupervised learning and machine learning with supervised learning along with theoretical guarantees about the learning process. We do not draw any such distinctions between the two in this work, and use the terms interchangeably.

Most often the "raw" input data is available as a set of vectors that may be grouped together into an input matrix. Given this raw data matrix, the goal of many data mining applications is to discover latent structure within it. For example, the goal might be to find clusters, to extract a dense network of strongly correlated input points, to find a subset of the features of the data, or to build regression models for it. Solving these problems is usually centered around matrix computations, and in this thesis we focus on typical data mining problems that can be viewed as matrix nearness problems.

By *matrix nearness* all we mean is that we are given an input matrix, and we wish to find an output matrix that is not only "near" the input, but also have certain desirable properties that are dictated by the needs of an application. The aim is to efficiently find such a nearby matrix that can be then employed in an application in lieu of the input matrix.

The nearness problems that we study take on two main forms. In the first form, we directly approximate the input matrix by another matrix, while in the second, we seek to learn some *parameterized model* of the input data. Both types of nearness problems are tied together by the common algorithmic techniques that are used for solving them. We describe these two scenarios a little more formally below.

Scenario one: Assume that the input data is available as the N vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$, which may be viewed as the columns of an $M \times N$ input matrix $\boldsymbol{A}$. Given $\boldsymbol{A}$, matrix nearness problems request a matrix $\hat{\boldsymbol{A}}$ that has cer-

tain desired characteristics such as nonnegativity, low-rank, sparsity, positive-definiteness, etc. Naturally, we would like to construct an $\hat{\boldsymbol{A}}$ that is as "near" $\boldsymbol{A}$ as possible so that that it may be used in place of $\boldsymbol{A}$ in the underlying application. Formally, this leads to the following optimization problem

$$\text{minimize} \quad \Delta(\hat{\boldsymbol{A}}, \boldsymbol{A}), \qquad \text{subject to } \hat{\boldsymbol{A}} \in \hat{\mathscr{A}}, \tag{1.1}$$

where $\Delta : \hat{\mathscr{A}} \times \mathscr{A} \to \mathbb{R}_+$ is a suitable distortion function that measures "closeness," and $\hat{\mathscr{A}}$ is the (feasible) set describing constraints on $\hat{\boldsymbol{A}}$. For most problems of interest $\hat{\mathscr{A}}$ is usually either a convex set or an appropriately structured set that can be parameterized easily, and Δ is either a distance function or a divergence measure (see §2.2.1).

Scenario two: Sometimes, the formulation (1.1) is not convenient to work with, particularly when one wishes to learn some parametric model from *or* of the input data, for e.g., a generative mixture model, a kernel function, or a distance metric. Assume the input data is given by $\boldsymbol{A}$ as before, and we wish to learn a model $\boldsymbol{M}_A$ for it (e.g., a probabilistic generative model). Often learning such a model proceeds by taking a given baseline model, say $\boldsymbol{M}_0$, and then computing an appropriately constrained model $\boldsymbol{M}_A$ chosen to lie within a class $\mathscr{M}$ of models. In symbols, this situation leads to the following optimization problem

$$\Delta(\boldsymbol{M}_0, \boldsymbol{M}_A), \quad \text{subject to } \boldsymbol{M}_A \in \mathscr{M}. \tag{1.2}$$

Note that the subscript A on the model $\boldsymbol{M}_A$ highlights the natural dependence of the model on the input data.

We now provide an overview of the specific problems that we study in this thesis. Section 1.1 below describes problems that fall under Scenario one, while Section 1.2 provides an overview of those that fall under Scenario two.

1.1 Low-rank Matrix Approximations

We begin our foray into matrix nearness problems by studying some important problems that assume the form (1.1). In particular, we study approximate low-rank matrix decompositions.[2] The aim of these approximations is to exploit the structure latent in the data to construct a low-parameter approximation to the input data. Consequently, one obtains benefits such as lower storage requirements, denoising, improved computational efficiency, and pattern discovery within the data.

Consider the input data $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$, where each $\boldsymbol{a}_i \in \mathbb{R}^M$. Usually both M and N are very large for data mining applications and we would like to somehow reduce the complexity of representing $\boldsymbol{A}$. A simple, but relatively powerful approach is to approximate each $\boldsymbol{a}_i$ by a linear combination of a small number of vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_K$, (where $K \ll \min(m, n)$) i.e.,

$$\boldsymbol{a}_n \approx \sum_k \boldsymbol{b}_k c_{kn}, \quad \text{for } 1 \leq n \leq N. \tag{1.3}$$

[2]Matrix decompositions are some of the most fundamental tools in matrix analysis, and they serve as key ingredients for numerous applications depending upon linear algebra. The most notable examples being the eigenvector decomposition, the SVD, LU, Cholesky, and QR decompositions. However, these decompositions are exact factorizations of the original data matrix, whereas we will concern ourselves with only approximate decompositions in this work.

The vectors $\boldsymbol{b}_k$ $(1 \leq k \leq K)$ may be thought of as "basis" vectors[3], and the accompanying c_{kn} values as "combining" coefficients. In matrix notation (1.3) may be written as

$$\boldsymbol{A} \approx \boldsymbol{BC},$$

where $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$ and $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N]$. As per (1.1), $\hat{\boldsymbol{A}} = \boldsymbol{BC}$, where $\hat{\mathscr{A}}$ can be specified by requiring $\boldsymbol{B} \in \mathcal{B}$, $\boldsymbol{C} \in \mathcal{C}$, for appropriate sets $\mathcal{B}$, and $\mathcal{C}$. Formally, (1.1) assumes the form

$$\min_{\boldsymbol{B},\boldsymbol{C}} \quad \Delta(\boldsymbol{A}, \boldsymbol{BC}), \quad \text{where } \boldsymbol{B} \in \mathcal{B}, \text{ and } \boldsymbol{C} \in \mathcal{C}. \tag{1.4}$$

The problem (1.4) is an instance of a low-rank matrix approximation problem because usually we constrain $\boldsymbol{B}$ and $\boldsymbol{C}$ to have rank much smaller than $\boldsymbol{A}$. The parameter sets $\mathcal{B}$ and $\mathcal{C}$ are used to describe additional constraints on $\boldsymbol{B}$ and $\boldsymbol{C}$ (e.g., normalization, nonnegativity, etc.).

Problem (1.4) is very general and provides a rich set of special cases through various choices of Δ and via imposition of different constraints on $\boldsymbol{B}$ and $\boldsymbol{C}$. Some simple examples are:

1. Singular Value Decomposition (SVD): Let $\Delta(\boldsymbol{A}, \boldsymbol{BC}) = \|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2$, $\mathcal{B} = \mathbb{R}^{M \times K}$, and $\mathcal{C} = \mathbb{R}^{K \times N}$, then (1.4) is solved by computing the truncated SVD of $\boldsymbol{A}$. Here, the rank K is the only constraint.

[3]These "basis" vectors can be alternatively called representative vectors, and each row of $\boldsymbol{B}$ may be thought of as a new "feature" obtained as a combination of the original features. Each representative vector $\boldsymbol{b}_k$ corresponds to some latent structure/factor within the data, a motivation obtained from *factor analysis*.

2. Non-negative Matrix Approximation (NNMA): Here we restrict $\mathcal{B} = \mathbb{R}_+^{M\times K}$ and $\mathcal{C} = \mathbb{R}_+^{K\times N}$. Specific versions depending on Δ are described in Section 1.1.1 below.

3. K-means Clustering: Let $\Delta(\boldsymbol{A}, \boldsymbol{BC}) = \|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2$, $\mathcal{B} = \mathbb{R}^{M\times K}$, and $\mathcal{C}$ is such that each column $\boldsymbol{c}_n$ of $\boldsymbol{C}$ equals a standard basis vector (all zeros except a unit entry in a single component corresponding to the cluster to which $\boldsymbol{a}_n$ belongs).

In addition to the examples given above, numerous other problems can be written by specializing (1.4). Furthermore, it is easy to extend the approximate decomposition to span several factors, i.e., we can generalize (1.4) to consider problems of the form

$$\min \quad \Delta(\boldsymbol{A}, \boldsymbol{B}_1\boldsymbol{B}_2\cdots\boldsymbol{B}_T), \quad \text{where } \boldsymbol{B}_t \in \mathcal{B}_t. \tag{1.5}$$

A particularly important example of (1.5) is given by co-clustering, for which $T = 3$ (see §2.9.6 for details).

Now we summarize the specific instances of (1.4) and (1.5) that are developed in this thesis. Our summary makes use of some technical terms that are only introduced in later chapters, but that should not be a hinderance to obtaining an overview of the problems.

1.1.1 Non-negative Matrix Approximation

We begin with non-negative matrix approximation (NNMA), where we wish to solve the following optimization problem:

$$\min \quad \Delta(\boldsymbol{A},\, \boldsymbol{BC}), \quad \text{subject to } \boldsymbol{B}, \boldsymbol{C} \geq 0. \tag{1.6}$$

NNMA was introduced by Paatero and Tapper [214], though it gained popularity only after the seminal work of Lee and Seung [174]. Both these papers studied the least-squares NNMA problem, where $\Delta = \|\cdot\|_F^2$, and the latter paper also introduced the KL-Divergence NNMA problem, where $\Delta(\boldsymbol{A},\, \boldsymbol{BC}) = \mathrm{KL}(\boldsymbol{A} \| \boldsymbol{BC})$. In Chapter 2 we develop methods based on iterative multiplicative updates for solving (1.6) and our methods subsume the well-known least-squares and KL-divergence based NNMA problems as special cases.

1.1.2 Incremental low-rank approximations

The generic techniques that we develop for solving (1.6) can be extended to solve (1.4) incrementally, wherein one computes $\boldsymbol{C}$ one column (and $\boldsymbol{B}$ one row) at a time. The incremental version of (1.4) may be written as

$$\min \quad \Delta(\boldsymbol{A},\, \boldsymbol{T} + \sigma \boldsymbol{u}\boldsymbol{v}^T), \tag{1.7}$$

subject to appropriate constraints on σ, $\boldsymbol{u}$, and $\boldsymbol{v}$. The matrix $\boldsymbol{T}$ denotes the current approximation to $\boldsymbol{A}$, and at each step we estimate σ, $\boldsymbol{u}$ and $\boldsymbol{v}$. In Chapter 3 we derive methods for solving (1.7) and some interesting variations

of it. Even though the formulation of (1.7) might appear to be too restrictive, it is quite powerful and works well in practice. Indeed, the well known singular value decomposition (SVD) is usually computed using an incremental procedure.

1.1.3 Decomposition Based Matrix Approximations

For the NNMA problem (1.6) we made no specific assumptions about the convexity of the distortion function Δ. In general, due to the nonlinear product term $\boldsymbol{BC}$, the function Δ fails to be convex over all the variables simultaneously. However, often Δ is individually convex in $\boldsymbol{B}$ and $\boldsymbol{C}$, whereby we can use an alternating minimization procedure, where each alternating step is a convex optimization problem. In Chapter 5 we revisit the low-rank approximation problem (1.4) and develop efficient methods for its solution. Our derivations yield as pleasant byproducts new algorithms for several interesting subproblems such as the non-negative least squares problem [171], and least ℓ_1-norm ($\min \|\boldsymbol{Bc} - \boldsymbol{a}\|_1$) problem.

1.2 Other Matrix Nearness Problems

We now turn our attention to special cases of (1.2), which are much less interrelated as compared to the different instantiations of (1.4). This difference arises because for (1.2) we seek to learn models/functions of the underlying data, rather than trying to directly approximate the data itself. The benefit of generalization here lies principally in: i) its unifying view which

suggests many new problems or even interesting variations to old ones, and ii) in its ability to exploit algorithmic techniques similar to those for (1.4) for solving the associated problems. The final two matrix nearness problems that we study in this thesis are summarized below.

1.2.1 Metric Nearness

Suppose that we have an input dissimilarity matrix $\boldsymbol{D}$ whose (i, j) entry represents some measure of dissimilarity between point $\boldsymbol{a}_i$ and point $\boldsymbol{a}_j$.[4] The goal of the *metric nearness* problem is to compute a matrix $\boldsymbol{M}$ whose entries specify a discrete metric over N (unspecified) points, and which is "near" $\boldsymbol{D}$. In the notation of (1.2), the base model $\boldsymbol{M}_0 = \boldsymbol{D}$, whereas the target model $\boldsymbol{M}_A = \boldsymbol{M}$. Explicitly, (1.2) may be specialized to

$$\min_{\boldsymbol{M}} \quad \Delta(\boldsymbol{D}, \boldsymbol{M}), \quad \text{subject to } \boldsymbol{M} \in \mathscr{M}_N, \tag{1.8}$$

where $\mathscr{M}_N$ denotes the convex cone of discrete metric matrices. We study this problem in Chapter 4, where we develop efficient algorithms for its solution and highlight its connection to the well-known all pairs shortest paths (APSP) problem.

[4]We would like to mention that for this problem, an explicit representation of the underlying points is unnecessary, i.e., the $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ can be left unspecified. We care about only the interpoint distances.

1.2.2 Parametric Mixture Modeling

The most familiar example of (1.2) is provided by the problem of parametric mixture modeling, where given the input data, one seeks to learn a probability density function $p(\boldsymbol{a}|\boldsymbol{\Theta})$ that could have generated it. Naturally, the problem is made simpler by assuming that the mathematical form of $p(\boldsymbol{a}|\boldsymbol{\Theta})$ is known and only the parameter matrix $\boldsymbol{\Theta}$ needs to be estimated. In parametric mixture modeling one assumes that the density of the observed variables is given by

$$p(\boldsymbol{a}|\boldsymbol{\Theta}) = \sum_{k=1}^{k=K} \alpha_k p(\boldsymbol{a}|\boldsymbol{\theta}_k), \tag{1.9}$$

where $\alpha_k \geq 0$, $\sum_k \alpha_k = 1$, and $p(\boldsymbol{a}|\boldsymbol{\theta}_k)$ is the k-th mixture component density. Now assume that we observe the i.i.d. samples $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$ so that the likelihood (expected incomplete data likelihood in EM parlance) is $\prod_n p(\boldsymbol{a}_n|\boldsymbol{\Theta})$. We wish to now estimate the parameters $\boldsymbol{\Theta}$ that are "most likely" to have generated the observed data. Thus we wish to compute $\boldsymbol{\Theta}$ so that the likelihood is maximized. It is easier to maximize the logarithm of the likelihood, and results in the following problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\Theta}} \quad \Big[\sum_{n=1}^{N} \log p(\boldsymbol{a}_n|\boldsymbol{\Theta}) = \sum_{n=1}^{N} \log\Big(\sum_{k=1}^{K} \alpha_k p(\boldsymbol{a}_n|\boldsymbol{\theta}_k)\Big)\Big]. \tag{1.10}$$

Problem (1.10) may be written as the matrix nearness problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\Theta}} \quad \mathrm{KL}(\mathbf{1}\mathbf{1}^T \| \boldsymbol{P}\boldsymbol{\alpha}\mathbf{1}^T) - N\big(\sum_n (\boldsymbol{P}\boldsymbol{\alpha})_n - 1\big), \tag{1.11}$$

where $\boldsymbol{P} = [p_{nj}]$ with $p_{nj} = p(\boldsymbol{a}_n|\boldsymbol{\theta}_j)$. Problem (1.11) essentially says that the goal of maximum-likelihood based mixture modeling is to compute a regularized maximum entropy estimate for the parameters. This view of maximum

likelihood by itself is not that useful—the interesting connection is via the method that is used for solving (1.11) (or (1.10) for that matter), namely the Expectation Maximization or EM procedure of Dempster et al. [71]. The EM method is derived following a technique identical to one of our approaches for solving the NNMA problem (see §2.3.1). Furthermore, since the EM procedure may be viewed as an alternating minimization procedure [61], it sheds new light on our NNMA algorithms. Finally, this connection suggests that algorithmic progress or variations made for EM can be exploited for NNMA and vice-versa, thereby underscoring the importance of viewing them both as matrix nearness problems.

In this thesis we solve two particular versions of (1.11), namely when the component density $p(\boldsymbol{a}_i|\boldsymbol{\theta}_k)$ is either the von Mises-Fisher or the Watson density [187]. These densities are fundamental to directional statistics, and they are important for large scale data in data mining and information retrieval, because for many applications absolute magnitude is less important than direction (or similarity). Directional distributions fit in naturally because they provide a generative approach for modeling such data. Unfortunately, parameter estimation for directional distributions is quite difficult, especially for high-dimensional data, as is typically encountered in data mining applications. In Chapter 6 we develop specialized algorithms for solving (1.11) with efficient methods for performing the associated parameter estimates that overcome the challenges posed by high-dimensionality.

1.3 Software

An important component of our research effort has been the development of efficient software for implementing the nearness algorithms that we have derived. We highlight the importance of good implementations for the problems that we study because naïve implementations can easily result in inefficient software. Furthermore, since data mining is an applications driven field, efficient software that permits the average user to apply our algorithms without having to go through either the engineering or the accompanying theory is very useful. Chapter 7 summarizes the software that accompanies this dissertation.

1.4 Related Work

Matrix decompositions have been relatively much better studied [106] than matrix approximations. Some of the earliest results in matrix approximation date back to Schmidt [249], who essentially proved the optimality of the rank-K SVD. Beginning with this venerable history, matrix nearness problems have come a long way (see the brief overview by Higham [127], for example). The Ph.D. thesis of Srebro [266] is most relevant to our work in spirit, though the problems he studies are different. For example, Srebro [266] studies elementwise weighted low-rank approximation and maximum margin matrix factorization, whereas we study the problems summarized above. We note that we do study one related problem, and that is elementwise weighted NNMA (see §2.9.7), which is a constrained optimization problem. We summarize be-

low several interesting matrix nearness/approximation problems relevant to data mining below.

1.4.1 Principal Components Analysis (PCA) and SVD

The most well-known low-rank matrix approximation is perhaps the truncated singular value decomposition [92, 106, 119, 249]. This decomposition retains the singular vectors and values corresponding to the largest K singular values of the matrix $\boldsymbol{A}$, yielding a rank-K approximation that is optimal with respect to any unitarily invariant norm [106].

The method of principal components analysis (PCA) [148] is an immediate consequence of the SVD. It projects the data onto the subspace spanned by the leading left singular vectors of the mean centered data (or equivalently onto the subspace spanned by the leading eigenvectors of the covariance matrix of the data), thereby capturing the directions of maximum variance within the data.

PCA has found wide applicability in numerous problems. Example applications include image processing and compression [236], signal processing systems [72], immunology [96], molecular dynamics [238], information retrieval, latent semantic indexing (LSI) [24] and analysis of gene expression data [281]. For other PCA approaches we refer the reader to [87]. More recently, PCA has been generalized in two ways. The first is Kernel PCA of Schölkopf et al. [252], which first forms the *kernel matrix* for the input data, performs eigendecomposition on it, and projects data onto these eigenvectors—by limiting

all computation to dot-products, Kernel PCA is able to exploit the "kernel trick" and achieve non-linear PCA. A second work [55], exploits the idea that traditional PCA is based on the assumption that the input data are Gaussian and PCA essentially maximizes the likelihood by finding parameters that lie in a low-dimensional space. Collins et al. [55] generalize PCA to the exponential family, where the data are assumed to be drawn according to some member of the exponential family whose natural parameters $\boldsymbol{\Theta}$ lie in some lower-dimensional space (indicated by writing $\boldsymbol{\Theta} = \boldsymbol{UV}$). Then, an appropriate Bregman divergence between the input data and $g(\boldsymbol{\Theta})$ is minimized, where $g(\boldsymbol{\Theta})$ denotes the matrix of expectation parameters. Their optimization problem is unconstrained and solved via alternating minimization over $\boldsymbol{U}$ and $\boldsymbol{V}$. Our work differs from [55] in that we have a constrained optimization problem, and there is no restriction to exponential families. Furthermore, we also provide actual algorithms to efficiently carry out the minimization (or descent) in the alternating steps.

1.4.1.1 Other Generalizations of PCA/SVD

Related to PCA is the generalization called PARAFAC that introduced by Harshman and Lundy [120], and it can be viewed as PCA for 3-dimensional data arrays. The methods of higher order SVD [68] and orthogonal tensor decompositions [157] deal with approximations of tensors in a manner analogous to the SVD. Similarly, Zhang and Golub [298] have considered approximating high-order tensors in a manner similar to SVD for matrices. However, the

fundamental ideas from matrices do not generally extend easily over to tensors because of difficult theoretical hurdles, for example, even the concept of (outer-product) rank is not well-defined for tensors [123].

Tensors are however finding increasing use. For example, applications to computer vision [243], face recognition [279], signal processing [28] exist, amongst many others. For a theoretical development of fundamental concepts such as rank, eigenvectors, etc. for tensors, we refer the reader to [69, 181].

1.4.2 Clustering and Co-Clustering

Clustering is one of the fundamental problems of data mining. There are a few books dedicated to the subject [121, 146, 199]. Clustering has been widely employed, for example there are applications in fields such as astrophysics [70], speech recognition and analysis [97, 159, 160], machine translation of text [59, 207], image processing [98, 151], text clustering [8, 12, 32, 77, 81], gene expression analysis [12, 20, 84, 93, 136, 257, 291].

Co-clustering can be traced back to an algorithm of Hartigan [122] who used the name *direct* clustering that became known as *block* clustering. Since then many other researchers have investigated co-clustering (and related methods such as subspace clustering). Co-clustering has been applied to text analysis and gene expression data analysis [13, 21, 42, 48, 49, 75, 83, 103, 155, 274]. For more references of co-clustering to problems in biology the reader is referred to the survey article by Madeira and Oliveira [184]. In addition, co-clustering has also been applied to simultaneous benefit segmentation and

market structuring [7].

Our matrix approximations yield relaxed solutions to both clustering and co-clustering, just as an SVD based method yields spectral clustering. Banerjee et al. [13] also study the relation of co-clustering and matrix approximations, essentially highlighting the fact that a co-clustering gives a low-parameter (and low-rank) approximation to the input data matrix. Initial work on studying the connection between clustering and low-rank approximations appeared in [275].

1.4.3 Independent Components Analysis (ICA)

The independent components problem has two main definitions. The general definition calls for the computation of a transforming matrix $\boldsymbol{W}$ so that, the transformation $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$, of an input vector $\boldsymbol{x}$, has components that are as statistically independent as possible. This definition is somewhat vague and hard for computation. An estimation-oriented definition assumes a generative model for the observed vector $\boldsymbol{x}$, and aims to compute it. That is, it assumes $\boldsymbol{x} \approx \boldsymbol{A}\boldsymbol{s}$, and tries to compute both the mixing matrix $\boldsymbol{A}$ and the independent components of $\boldsymbol{s}$. ICA can be applied to blind source separation, in which the input $\boldsymbol{x}$ represents linearly mixed source signals, and $\boldsymbol{s}$ represents the uncorrupted source signals. ICA has also been applied to feature extraction, in which each s_i is the coefficient of the i-th feature in the observed data vector $\boldsymbol{x}$. An excellent survey is given in the article by Hyvärinen [139] and a more leisurely development can be found in [137].

A version of ICA that requires the sources $\boldsymbol{s}$ to be nonnegative was introduced under the name of nonnegative ICA by Plumbley [229]. A related paper discusses solving the nonnegative ICA problem by using nonnegative PCA [209], that is a special case of the nonlinear PCA algorithm, with a rectification nonlinearity. A further level of sophistication was added by Cardoso [44], who made use of fourth order tensors for ICA estimation, however, the resulting methods are computationally intensive.

Discussion

Hyvärinen [139] calls methods such as PCA and factor analysis as second-order methods, because these methods proceed essentially by making use of the information contained in the covariance matrix of an observed (or input) variable for finding a suitable representation. They point out that these methods aim to find *faithful* representations of the input variable as they try to minimize a distortion or error criterion, whereas higher-order methods such as ICA try to find a *meaningful* representation. This viewpoint is interesting to consider, because we concentrate on the both faithfulness and interpretability when performing NNMA.

1.4.4 Generalized Linear Models

Generalized Linear Models were originally introduced by Nelder and Wedderburn [205], and they have been investigated intensely since then. GLMs extend the notion of linear regression, where it is assumed that the dependent

variable, say $\boldsymbol{x}$ is equal to a linear combination $\boldsymbol{Z}^T\boldsymbol{\beta}$ plus a normally distributed error $\boldsymbol{\epsilon}$ so that

$$\boldsymbol{x} = \boldsymbol{Z}^T\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

In GLMs we assume that $\boldsymbol{x}$ is distributed according to a member of the exponential family. The GLM then is

$$\boldsymbol{x} = g(\boldsymbol{Z}^T\boldsymbol{\beta}) + \boldsymbol{\epsilon},$$

where the task is to estimate $\boldsymbol{\beta}$ given the *link* function g, the observed data $\boldsymbol{x}$ and the combination matrix $\boldsymbol{Z}$. The error $\boldsymbol{\epsilon}$ is usually assumed to have mean 0. The quantity $g(\boldsymbol{Z}^T\boldsymbol{\beta})$ approximates the expectation parameter of the exponential model being employed. The interested reader is referred to published literature for further details, e.g., [88, 191].

A further generalization to GLMs was made by Gordon [107] who introduced a model called Generalized^2 Linear^2 Models ((GL)^{2}Ms). This new model allows one to introduce link functions to allow non-linear combinations of transformations of the underlying components. Specifically (GL)^{2}Ms decompose an input matrix $\boldsymbol{X}$ as

$$\boldsymbol{A} \approx f(g(\boldsymbol{B})h(\boldsymbol{C})).$$

These models include as special cases PCA, exponential-family PCA and GLMs. Note that we can also consider such models in our low-rank approximation framework, but our work differs by the critical inclusion of constraints on $\boldsymbol{B}$ and $\boldsymbol{C}$, that make the associated optimization problems much harder.

As before, we also provide efficient algorithms for performing the associated approximations.

1.4.5 Miscellaneous

Some other related matrix approximation problems are listed below.

1. Vector Quantization seeks to approximate the input data by the nearest prototype vector. That means, the matrix $\boldsymbol{B}$ is approximated from the input data, and for each input vector $\boldsymbol{a}_n$, the column $\boldsymbol{c}_k$ contains a single unit element that allows it to pick out the best prototype from $\boldsymbol{B}$. Observe the similarity to a hard-clustering procedure.

2. Semi-discrete decomposition [158] approximates the input matrix $\boldsymbol{A}$ by the sum $\sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ (note similarity to SVD), where the components of $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are constrained to lie set $\{-1, 0, 1\}$. Naturally, this is a combinatorial optimization problem, though relaxations via the SVD with thresholding are natural.

3. The Discrete Basis Problem [197] seeks to find binary matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ that approximate a binary input matrix $\boldsymbol{A}$, so that $\boldsymbol{A} \approx \boldsymbol{B} \circ \boldsymbol{C}$, where $\circ$ denotes the boolean product. This is again a hard combinatorial problem, and some relaxed solutions with thresholding can be pursued as for semi-discrete decompositions.

4. Linear discriminant analysis: Just as PCA seeks to project the data onto the subspace where it is maximally decorrelated, linear discriminant

analysis seeks a projection that maximally "discriminates" or separates the data [91]. It has also been kernelized [198].

5. Probabilistic latent semantic indexing (PLSI) [130] gives a statistical grounding to the SVD based procedure of latent semantic indexing (LSI). However, it can be shown that PLSI and NNMA share local optima, and are almost the same (the difference between them is that of an extra normalization matrix that PLSI computes, but NNMA does not).

6. Sparse code shrinkage [138, 140] aims to find a sparse combining vector $\boldsymbol{c}$ so that $\boldsymbol{a} \approx \boldsymbol{B}\boldsymbol{c}$. It can also be related to ICA if one demands sparsity in addition to independence in the components of $\boldsymbol{c}$.

7. Local feature analysis [225] can be seen as a sparse or local version of PCA. Sparse PCA itself is quite popular, see for e.g., [63, 302].

8. Nonlinear dimensionality reduction methods [242, 272, 290].

9. Problems such as the Euclidean distance matrix completion problem [64] and the kernel matrix completion problem [278].

10. Metric learning: Suppose we have N points $\{\boldsymbol{a}_i, \ldots, \boldsymbol{a}_N\}$ as before. Assume that we are given pairwise constraints that require the interpoint distance between "similar" points to be small and between "dissimilar" points to be large. The task of metric learning is to find a (parameterized) distance metric which satisfies these pairwise distance constraints [65, 105, 251, 256, 285, 292]

Chapter 2

Non-negative Matrix Approximation

Note: Most of the material of this chapter is based on our work [79, 265]

2.1 Introduction

The first matrix nearness problem that we study in this thesis is the non-negative matrix approximation (NNMA) problem, which was briefly introduced in Section 1.1.1. NNMA is a powerful matrix decomposition technique that has been widely applied to numerous applications in fields as diverse as text analysis, image processing, chemometrics, and bioinformatics, and continues to find newer uses. This broad applicability coupled with the rich variety of subproblems that it entails, make NNMA a particularly attractive matrix nearness problem for us to study, and we devote this chapter to this pursuit.

The discovery of latent information within data has been the subject of considerable analysis for several decades, especially in domains that apply statistical methods to analyze their problems. With the advent of machine learning and data mining, the search for structure or information from within the data has taken on a new perspective—the data sets are large and often very sparse, the data dimensionality is high, frequently running into tens or

hundreds of thousands of features, and naturally, the data are usually noisy.

A first attempt to tackle some of these problems is the method of *dimensionality reduction*, where one projects the high-dimensional input data into a lower dimensional space. This step has several benefits: i) it can reveal latent structure within the data, ii) it reduces the data complexity leading to a more efficient representation with accompanying storage and computational benefits, and iii) it helps to denoise the data. The method of Principal Components Analysis (PCA) is by far the most well-known dimensionality reduction method, and it proceeds by computing a truncated Singular Value Decomposition (SVD) of mean centered data (which essentially leads to a minimization of Gaussian noise).

Despite its advantages, PCA suffers from some practical difficulties, which become more pronounced for data mining applications. First, it produces low-rank projections that are dense, even if the original data is very sparse, which can be a severe drawback for data mining problems where large sparse data sets are the norm. Second, from a practitioners perspective, PCA produces results that are not always amenable to interpretation. For example, suppose that our input data consists of inherently non-negative quantities such as frequency counts, color intensities, chemical concentrations, joint probabilities, etc. For such data PCA produces representations that contain negative entries, which lack physical meaning and defy useful interpretation.

To address these practical concerns, researchers were led to consider dimensionality reduction problems that respect the non-negativity of the data,

and this is where NNMA comes to the forefront. Furthermore, the nonnegativity requirement of NNMA automatically leads to sparse representations too. This is however, not surprising, because in order to have a low reconstruction error, several variables must be set to zero as no benevolent cancellations due to opposing signs can occur.

2.2 Problem formulation

Let us now consider the NNMA problem more formally. Given a nonnegative matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}_+^{M \times N}$ as input, the goal of NNMA is to construct a low-rank approximation of the form $\boldsymbol{A} \approx \boldsymbol{BC}$, where $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_K] \in \mathbb{R}_+^{M \times K}$ and $\boldsymbol{C} = [\boldsymbol{c}_1, ..., \boldsymbol{c}_N] \in \mathbb{R}_+^{K \times N}$ are themselves nonnegative. For computing $\boldsymbol{B}$ and $\boldsymbol{C}$, we solve the following optimization problem

$$\begin{aligned} \min \quad & \Delta(\boldsymbol{A},\, \boldsymbol{BC}), \\ \text{subject to } & \boldsymbol{B}, \boldsymbol{C} \geq 0, \end{aligned} \tag{2.1}$$

where the function Δ is an appropriate divergence (distance or distortion) function. The number of distance measures that can be considered is staggeringly large; some useful reference sources that we can recommend to the reader are the excellent paper of Bauschke and Borwein [16], and the two recent books [74, 219]. For a quick reference, we present some relevant background about divergence measures in §A.1.3.

We begin by studying Problem (2.1) when Δ is a *Bregman divergence*, and later present extensions to some other interesting divergence functions. Below we summarize some important properties about these divergences (see

§A.1.3 for more details).

2.2.1 Bregman divergences

A Bregman divergence is generated by an underlying function φ, which is closed, strictly convex, and proper on $S \subseteq \mathbb{R}$, where $\mathrm{ri}(S) \neq \emptyset$ (ri denotes the relative interior). If φ has a continuous first derivative on $\mathrm{ri}(S)$, the corresponding *Bregman divergence* $D_\varphi : S \times \mathrm{ri}(S) \to \mathbb{R}_+$ is defined as (see [45])

$$D_\varphi(x; y) \stackrel{\text{def}}{=} \varphi(x) - \varphi(y) - \varphi'(y)(x - y). \tag{2.2}$$

It is easy to see that (2.2) is the first-order Taylor series expansion of $\varphi(x)$, and we may intuitively view D_φ as a measure of departure from linearity of the function φ. Bregman divergences play an important role in convex optimization [37, 38, 45] and probabilistic modeling [14, 15], and learning theory [54, 283], largely because of many useful properties that they possess. For example, they are nonnegative, convex in the first argument and zero if and only if $x = y$. However, these divergences are not true distance metrics, because they are almost always asymmetric, i.e., $D_\varphi(x; y) \neq D_\varphi(y; x)$. The well known squared Euclidean distance and the information theoretic KL-Divergence (unnormalized) are two particular Bregman divergences, generated respectively by $\varphi(x) = \frac{1}{2}x^2$, and $\varphi(x) = x \log x$. Since a conic combination of two Bregman divergences is also a Bregman divergence, we may extend the definition (2.2) to matrix arguments, so that

$$D_\varphi(\boldsymbol{X}; \boldsymbol{Y}) \stackrel{\text{def}}{=} \sum_{ij} D_\varphi(x_{ij}; y_{ij}). \tag{2.3}$$

We call (2.3) a *separable* divergence because it is an elementwise function of the matrix entries. Non-separable Bregman divergences over matrices can also be defined by appropriately composing the convex function φ with the eigenvalue map [178]. For the duration of this chapter, we will concern ourselves only with separable divergences. See [80, 162] for some work on matrix nearness problems with non-separable Bregman divergences.

2.2.2 The Problems

Given the definition (2.3) we finally set up the following two main optimization problems as instantiations of (2.1),

$$\min_{\boldsymbol{B},\boldsymbol{C}\geq 0} \quad D_\varphi(\boldsymbol{BC};\boldsymbol{A}), \quad \text{and} \tag{2.4}$$

$$\min_{\boldsymbol{B},\boldsymbol{C}\geq 0} \quad D_\varphi(\boldsymbol{A};\boldsymbol{BC}). \tag{2.5}$$

We consider both versions (2.4) and (2.5), since as previously mentioned Bregman divergences are usually asymmetric, whereby each version leads to interesting algorithms with differing characteristics. In the sequel, we derive efficient algorithms for finding solutions to both these problems, and we also demonstrate several important special cases to demonstrate the power of our approach.

2.2.3 General Approach for Solution

Before we embark upon specific solutions of (2.4) and (2.5) it will be worthwhile to look at the generic approach that underlies our algorithms, not

only for these problems, but also for several other structured matrix approximation problems taking the form (1.4).

Due to the non-linear product term $\boldsymbol{BC}$ in the approximations, the resulting problems are non-convex. Oftentimes the objective function might be individually convex in either $\boldsymbol{B}$ or $\boldsymbol{C}$, in which case we can provide better theoretical guarantees about the resulting algorithm. However, we do not always depend on convexity, especially for Problem (2.5). For this problem invoke the following generic iterative alternating descent scheme:

ALGORITHM 2.1: Generic Alternating Descent Procedure

ALTERNATING_DESCENT($\boldsymbol{A}$, K)
Input: $\boldsymbol{A}$, K
Output: $\boldsymbol{B}$, $\boldsymbol{C}$
{Initialization}
Let $t \leftarrow 0$; Initialize $\boldsymbol{B}^0$ and/or $\boldsymbol{C}^0$
repeat
  $(*)$ Compute $\boldsymbol{B}^{t+1}$ so that $\Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1}\boldsymbol{C}^t) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t\boldsymbol{C}^t)$
  $(**)$ Compute $\boldsymbol{C}^{t+1}$ so that $\Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1}\boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1}\boldsymbol{C}^t)$
  $t \leftarrow t + 1$
until *convergence*.

Algorithm 2.1 ensures monotonic descent in $\Delta(\boldsymbol{A}, \boldsymbol{BC})$ by construction, and since $\Delta \geq 0$ is bounded below and its domain can be considered to be compact, the sequence of objective function values $\{\Delta(\boldsymbol{A}, \boldsymbol{B}^t\boldsymbol{C}^t)\}$ has a limiting value Δ^*. However, note that this convergence is not sufficient to guarantee the convergence of $\{\boldsymbol{B}^t, \boldsymbol{C}^t\}$ to a limit point $\{\boldsymbol{B}^*, \boldsymbol{C}^*\}$, primarily because any such limit point is part of a continuum of limit points, since $\boldsymbol{BC} = (\boldsymbol{BT})(\boldsymbol{T}^{-1}\boldsymbol{C})$, for any invertible matrix $\boldsymbol{T}$. In particular, for some pos-

itive diagonal matrix $\boldsymbol{T}$, its inverse is also a positive diagonal matrix, so that if $\{\boldsymbol{B}^*, \boldsymbol{C}^*\}$ is a (non-negative) fixed point of the algorithm, so is $\{\boldsymbol{B}^*\boldsymbol{T}, \boldsymbol{T}^{-1}\boldsymbol{C}^*\}$.

Even if we select some fixed point, say $\{\boldsymbol{B}^*, \boldsymbol{C}^*\}$, it is not always possible to show that this fixed point is a stationary point of (2.1), leave alone arguing about its local-optimality. If we, however, replace the descent steps marked by $(*)$ and $(**)$ in Algorithm 2.1 by *exact minimization*, i.e., we perform the updates

$$\begin{aligned}\boldsymbol{B}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{B}\geq 0} \ \Delta(\boldsymbol{A},\, \boldsymbol{B}\boldsymbol{C}^t),\\ \boldsymbol{C}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{C}\geq 0} \ \Delta(\boldsymbol{A},\, \boldsymbol{B}^{t+1}\boldsymbol{C}),\end{aligned}$$

then usually one can ensure convergence to a local-minimum. We will take up this thread again in Chapter 5 and for now focus on algorithms that ensure only descent, since they are simpler and often faster than those performing exact minimization sub-steps [153].

2.3 Descent Algorithms for (2.4)

In this section we develop generic methods for solving (2.4), which essentially are appropriate instantiations of the descent procedure given as Algorithm 2.1.

Since the divergences that we treat are separable (elementwise functions of the entire matrix), we can compute $\boldsymbol{C}$ by solving n separate subproblems. Thus, we illustrate our method for an arbitrary column $\boldsymbol{c}$ of $\boldsymbol{C}$. Clearly, the approach can be followed symmetrically for rows of $\boldsymbol{B}$, so we omit it for brevity.

Let $\boldsymbol{a}$ be the column of $\boldsymbol{A}$ corresponding to $\boldsymbol{c}$. For a fixed value of the matrix $\boldsymbol{B}$, the subproblem is

$$\min_{\boldsymbol{c} \geq 0} \quad F(\boldsymbol{c}) = D_{\varphi}(\boldsymbol{B}\boldsymbol{c}; \boldsymbol{a}). \tag{2.6}$$

Since Bregman divergences are strictly convex in their first argument, we could minimize $F(\boldsymbol{c})$ (subject to $\boldsymbol{c} \geq 0$) using an off-the-shelf convex optimization software. However, such an approach can be computationally expensive; in addition if one must implement the convex optimization software oneself, it entails a considerably greater implementation burden. Thus, if we are willing to tradeoff precision for speed while solving (2.6) we can switch to a simple descent procedure. This approach is not uncommon, as the next section will soon show. We alert the reader to the fact that even though we are performing mere descent and not exact minimization at each sub-step, it is hard to say in general how this affects the overall solution, because the original problem is non-convex. One would expect exact minimization to lead to overall better local minima than mere descent would entail, but this cannot be usually theoretically guaranteed. We return to exact minimization in Chapter 5.

2.3.1 Multiplicative Updates via Auxiliary Functions

As per the structure of Algorithm 2.1, our goal is to construct a descent procedure so that $F(\boldsymbol{c}^{t+1}) \leq F(\boldsymbol{c}^t)$. We seek to obtain multiplicative updates, so that it is easy to ensure non-negativity of $\boldsymbol{c}$. For subproblem (2.6) this

amounts to updating $\boldsymbol{c}$ as

$$c_p \leftarrow \eta_p c_p, \quad \text{where } \eta_p \in \mathbb{R}_+, \quad \text{and } 1 \leq p \leq K. \tag{2.7}$$

We seek elementwise updates for $\boldsymbol{c}$ in (2.7), however, in the objective function $F(\boldsymbol{c})$ the components of $\boldsymbol{c}$ are "coupled" together via the matrix $\boldsymbol{B}$. Hence, we needs to somehow decouple the components of $\boldsymbol{c}$ in order to obtain elementwise updates. We construct certain auxiliary functions below that achieve this decoupling and help us derive iterative descent algorithms. The idea of auxiliary functions has been previously used most notably in the construction of the Expectation Maximization (EM) algorithm [71]. In the context of NNMA they were also used by Lee and Seung [172] for obtaining their algorithms.

Definition 2.1 (Auxiliary function)**.** A function $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ is called an auxiliary function for $F(\boldsymbol{c})$ if:

1. $G(\boldsymbol{c}, \boldsymbol{c}) = F(\boldsymbol{c})$ for all $\boldsymbol{c} \geq 0$, and
2. $G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) \geq F(\boldsymbol{c})$ for all $\boldsymbol{c}, \tilde{\boldsymbol{c}} \geq 0$.

Suitably constructed auxiliary functions turn out to be useful for deriving iterative descent algorithms for minimizing $F(\boldsymbol{c})$ primarily due to the following lemma.

Lemma 2.2 (Iterative minimization)**.** *If $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ is an auxiliary function for $F(\boldsymbol{c})$, then F is non-increasing under the update*

$$\boldsymbol{c}^{t+1} = \operatorname*{argmin}_{\boldsymbol{c} \geq 0} G(\boldsymbol{c}, \boldsymbol{c}^t). \tag{2.8}$$

Proof. $F(\boldsymbol{c}^{t+1}) \leq G(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t) \leq G(\boldsymbol{c}^t, \boldsymbol{c}^t) = F(\boldsymbol{c}^t)$. The first inequality and the last equality follow from Definition 2.1, whereas the second inequality follows from (2.8). □

Note that forming the auxiliary function $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ is analogous to the E-step of EM, whereas the optimization (2.8) is analogous to the M-step. Given some initial $\boldsymbol{c}^0 \geq 0$, an iterative application of Lemma 2.2 yields a sequence $\{\boldsymbol{c}^t\}$ such that $F(\boldsymbol{c}^0) \geq F(\boldsymbol{c}^1) \geq \cdots \geq F(\boldsymbol{c}^{t+1})$. Since F is bounded below by 0 and the underlying domain is compact, the sequence $\{F(\boldsymbol{c}^t\}$ has a limit $F(\boldsymbol{c}^*)$. Without further restrictions, it is not always easy to prove that this limit point is a stationary point of (2.6), or even that $\{\boldsymbol{c}^t\} \rightarrow \boldsymbol{c}^*$.

Remarks. Our derivation is based on auxiliary functions, just like its more famous cousin, the EM method, whereby we could borrow proof techniques from the EM algorithm for analyzing the convergence properties of our approach. EM has witnessed a tremendous amount of research effort both from a theoretical practical perspective [192]. As noted by Wu [289] (and others), the original convergence proof of Dempster et al. [71] was flawed. Subsequently by appealing to an appropriate adaption of Zangwill's general global convergence theorem [295], Wu analyzed the convergence properties of EM. However, several assumptions made by Wu turn out to be difficult to assess in general, thereby limiting the application of his approach to a similar attack on the convergence analysis of (2.4). Finally, Tseng [276] provides a clean and rigorous analysis of the convergence properties of EM by appealing to the proximal

minimization [45, 237] view of the EM algorithm. However, even Tseng [276] acknowledges the difficulty of removing the most common assumption made by convergence proofs of EM, namely, that the optimal solution lies strictly in the interior of the constraint set. This is a huge drawback for NNMA, because due to the nature of the non-negative constraints, a large number of components of the solution actually lie on the boundary. This drawback makes it impossible to provide a simple analysis of the convergence properties without making the (unrealistic) assumption about the solution lying in the interior. A rigorous analysis will be highly non-trivial, and is deferred to future work.

2.3.2 Constructing the auxiliary function

Lemma 2.3 below shows how to easily construct an auxiliary function for $F(\boldsymbol{c})$. Naturally, there can be infinite variety in the types of auxiliary functions that we could construct, and we present one simple choice below.

Lemma 2.3 (Auxiliary function)**.** *The function*

$$G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = \sum_{ij} \lambda_{ij} \varphi\left(\frac{c_j}{\tilde{c}_j}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i\right) - \left(\sum_i \varphi(a_i) + \nabla\varphi(a_i)\big((\boldsymbol{B}\boldsymbol{c})_i - a_i\big)\right), \quad (2.9)$$

with $\lambda_{ij} = (b_{ij}\tilde{c}_j)/(\sum_l b_{il}\tilde{c}_l)$*, is an auxiliary function for* $F(\boldsymbol{c})$ *(as defined in Eq. 2.6).*

Proof. It is easy to verify that $G(\boldsymbol{c}, \boldsymbol{c}) = F(\boldsymbol{c})$. Since $\sum_j \lambda_{ij} = 1$ and $\lambda_{ij} \geq 0$

(b_{ij} and $\tilde{c}_j \geq 0$), using the convexity of φ we find that

$$\begin{aligned} F(\boldsymbol{c}) = &\quad D_\varphi(\boldsymbol{B}\boldsymbol{c}; \boldsymbol{a}) \\ = &\quad \sum_i \varphi\Big(\sum_j b_{ij}c_j\Big) - \Big(\sum_i \varphi(a_i) + \nabla\varphi(a_i)\big((\boldsymbol{B}\boldsymbol{c})_i - a_i\big)\Big) \\ \leq &\quad \sum_{ij} \lambda_{ij}\varphi\Big(\frac{b_{ij}c_j}{\lambda_{ij}}\Big) - \Big(\sum_i \varphi(a_i) + \nabla\varphi(a_i)\big((\boldsymbol{B}\boldsymbol{c})_i - a_i\big)\Big) \\ = &\quad G(\boldsymbol{c}, \tilde{\boldsymbol{c}}). \end{aligned}$$

□

Note that to obtain our auxiliary function we manipulated only the first term of $F(\boldsymbol{c})$. Contributions from the other terms could also be involved, yielding different auxiliary functions. Furthermore, for certain classes of functions φ, we could exploit other bounding inequalities to obtain different auxiliary functions (for e.g., see [54]).

Now that we have $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$, only solving (2.8) remains, for which we must minimize $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ w.r.t. $\boldsymbol{c}$. Let $\nabla\varphi(\boldsymbol{x})$ denote the gradient vector $[\nabla\varphi(x_i)]$. Then, the partial derivative of $\partial G/\partial c_p$ is

$$\begin{aligned} \frac{\partial G}{\partial c_p} &= \sum_i \lambda_{ip}\nabla\varphi\Big(\frac{b_{ip}c_p}{\lambda_{ip}}\Big)\frac{b_{ip}}{\lambda_{ip}} - \sum_i b_{ip}\nabla\varphi(a_i) \\ &= \sum_i b_{ip}\nabla\varphi\Big(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i\Big) - (\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a}))_p. \end{aligned} \tag{2.10}$$

If the solution of (2.10) turns out to be non-negative (i.e., $\partial G/\partial c_p = 0$ has a non-negative solution c_p), then we are done, else we will need to perform a constrained minimization of $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$. Fortunately, for a large variety of objective functions guaranteeing non-negativity of c_p is easy (see Table 2.1 for examples). The final problem that remains is to see whether we can obtain an

analytic solution to $\partial G/\partial c_p = 0$ or not. Solving this equation is analogous to the M-step in an EM procedure. Just as for a large class of probability distributions, the M-step in an EM procedure can be efficiently solved to yield closed form parameter updates, the equation $\partial G/\partial c_p = 0$ can be solved analytically for the following two broad classes of functions:

1. Multiplicative $\nabla\varphi$, i.e., $\nabla\varphi(xy) = \nabla\varphi(x)\nabla\varphi(y)$,

2. Additive $\nabla\varphi$, i.e., $\nabla\varphi(xy) = \nabla\varphi(x) + \nabla\varphi(y)$.

Method	$\varphi(x)$	$\nabla\varphi(x)$	$\nabla\varphi^{-1}(x)$
Update (2.12)	$\varphi(x) = \frac{1}{r}x^r$	$\nabla\varphi(x) = x^{r-1}$ (multiplicative)	$\nabla\varphi^{-1}(x) = x^{\frac{1}{r-1}}$
Simplify (2.10)	$ax\log x + bx$	$\nabla\varphi(x) = a\log x + b$ (additive)	$\nabla\varphi^{-1}(x) = e^{\frac{x-b}{a}}$

Table 2.1: Examples of functions for which (2.10) can be solved in closed form. For other cases, (2.10) can be easily solved using a nonlinear root-finder.

Just as for the EM algorithm, sometimes the optimization step (2.8) cannot be easily computed. In such cases we might just settle for a *generalized* update, wherein $\boldsymbol{c}^{t+1}$ is computed to ensure

$$G(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t) \leq G(\boldsymbol{c}^t, \boldsymbol{c}^t). \tag{2.11}$$

This approach is motivated by the success of the Generalized EM (GEM) procedure of Dempster et al. [71]. This procedure is called generalized because the iterative scheme based on (2.8) is a special case, and it will prove to be useful for dealing with regularized NNMA problems as studied in Section 2.4.

Multiplicative $\nabla\varphi$. For this case we can solve $\partial G/\partial c_p = 0$ as follows,

$$\begin{aligned}
\frac{\partial G}{\partial c_p} &= \sum_i b_{ip} \nabla\varphi\left(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i\right) - (\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a}))_p = 0, \\
&\sum_i b_{ip} \nabla\varphi\left(\frac{c_p}{\tilde{c}_p}\right)\nabla\varphi\big((\boldsymbol{B}\tilde{\boldsymbol{c}})_i\big) - (\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a}))_p = 0, \\
&\nabla\varphi\left(\frac{c_p}{\tilde{c}_p}\right)[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p = [\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p \\
&\nabla\varphi\left(\frac{c_p}{\tilde{c}_p}\right) = \frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p}{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}.
\end{aligned}$$

Thus, we obtain the update

$$c_p \leftarrow \ \tilde{c}_p \cdot (\nabla\varphi)^{-1}\Big(\frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p}{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}\Big). \tag{2.12}$$

We can compute the updates for $\boldsymbol{B}$ one row at a time in a similar fashion to obtain

$$b_p \leftarrow \tilde{b}_p \cdot (\nabla\varphi)^{-1}\Big(\frac{[\nabla\varphi(\boldsymbol{a}^T)\boldsymbol{C}^T]_p}{[\nabla\varphi(\tilde{\boldsymbol{b}}^T\boldsymbol{C})\boldsymbol{C}^T]_p}\Big). \tag{2.13}$$

Additive $\nabla\varphi$. Assume for simplicity that $\nabla\varphi(xy) = \nabla\varphi(x) + \nabla\varphi(y)$. We solve $\partial G/\partial c_p = 0$ as follows,

$$\begin{aligned}
\frac{\partial G}{\partial c_p} &= \sum_i b_{ip} \nabla\varphi\left(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i\right) - (\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a}))_p = 0, \\
&\sum_i b_{ip}\left(\nabla\varphi\big(\frac{c_p}{\tilde{c}_p}\big) + \nabla\varphi\big((\boldsymbol{B}\tilde{\boldsymbol{c}})_i\big)\right) - (\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a}))_p = 0, \\
&\nabla\varphi\big(\frac{c_p}{\tilde{c}_p}\big)[\boldsymbol{B}^T\mathbf{1}]_p = [\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p - [\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p \\
&\nabla\varphi\big(\frac{c_p}{\tilde{c}_p}\big) = \frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p - [\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}{[\boldsymbol{B}^T\mathbf{1}]_p}.
\end{aligned}$$

Thus, we obtain the update

$$c_p \leftarrow \tilde{c}_p \cdot (\nabla\varphi)^{-1}\left(\frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p - [\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}{[\boldsymbol{B}^T\mathbf{1}]_p}\right). \tag{2.14}$$

2.3.3 Remarks and Observations

1. When φ is a convex function of Legendre type, then $(\nabla\varphi)^{-1}$ can be obtained by the derivative of the conjugate function φ^* of φ, i.e., $(\nabla\varphi)^{-1} = \nabla\varphi^*$ [237].

2. Since the Frobenius norm is a symmetric Bregman divergence (generated by $\varphi(x) = \frac{1}{2}x^2$), it comes as no surprise that (2.14) & (2.13) coincide with the Frobenius norm NNMA updates derived by Lee and Seung [172].

3. The auxiliary functions derived above depend only on the fact that Bregman divergences are convex in their first argument and that $\boldsymbol{B}, \boldsymbol{c} \geq 0$. Therefore, for minimizing a distortion measure $D(\boldsymbol{Bc}, \boldsymbol{a}) = \sum_i D_i((\boldsymbol{Bc})_i, a_i)$, where each individual distortion function D_i is convex in its first argument, we may use the following general approach:

   (a) Let $\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}$.

   (b) $D(\boldsymbol{Bc}, \boldsymbol{a}) = \sum_i D_i((\boldsymbol{Bc})_i, a_i) \leq \sum_{ij} \lambda_{ij} D_i(\frac{b_{ij}c_j}{\lambda_{ij}}, a_i) = G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$.

   (c) Optimize $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ w.r.t. each component c_p by setting its derivative to zero and solving

$$\sum_i b_{ip}\, \nabla D_i\big(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i, a_i\big) = 0.$$

Our auxiliary function method from the previous section can be obtained by following the above procedure (with minor modifications) with $D \equiv D_\varphi$. Thus, in fact, our method is extensible to a large variety of convex loss functions, and is not limited to Bregman divergences.

4. As a part of step 3(a) we can select some other suitable set of convex coefficients, for e.g., to use an annealing approach we can select

$$\lambda_{ij} = \frac{(b_{ij}\tilde{c}_j)^T}{\sum_l [b_{il}\tilde{c}_l]^T}, \tag{2.15}$$

where T denotes the so-called *computational temperature*. An even more general approach is provided by the following choice of convex coefficients

$$\lambda_{ij} = \frac{\gamma(b_{ij}\tilde{c}_j)}{\sum_l \gamma(b_{ij}\tilde{c}_j)}, \tag{2.16}$$

for a given non-negative function γ. Both of these choices lead to auxiliary functions for which we can only assure $G(\boldsymbol{c}, \boldsymbol{c}) \leq F(\boldsymbol{c})$ for all $\boldsymbol{c}$, leading to generalization of the auxiliary function approach. See [117] for related ideas in context of the EM algorithm.

5. **Complexity.** The update (2.12) for each column of the matrix $\boldsymbol{C}$ can be implemented in time $O(KM)$. Similarly the update for each row of $\boldsymbol{B}$ can be implemented to take $O(KN)$ time (assuming the computation of the gradient of φ and its inverse takes constant time). Thus, the overall algorithm runs in time $O(TKMN)$, where T is the number of iterations. Often, one can take advantage of the sparsity of $\boldsymbol{A}$ to decrease the running time to $O(TKnz(\boldsymbol{A}))$, where $nz(\boldsymbol{A})$ denotes the number of non-zeros in the matrix $\boldsymbol{A}$.

2.4 Regularized Version of Problem (2.4)

Sometimes one wishes to further control the factors $\boldsymbol{B}$ and $\boldsymbol{C}$ while computing a non-negative approximation. For e.g., one might restrict the size of the individual entries of $\boldsymbol{B}$ or $\boldsymbol{C}$ by imposing a penalty on their norms. This not only fosters even sparser solutions, but also helps to potentially counter ill-conditioning. The regularized NNMA problem is

$$\min_{\boldsymbol{B},\boldsymbol{C}\geq 0} \quad D_{\varphi}(\boldsymbol{BC};\boldsymbol{A}) + \alpha(\boldsymbol{B}) + \beta(\boldsymbol{C}), \tag{2.17}$$

where α and β are appropriate (non-negative, differentiable, and for now, separable) functions. For e.g., $\alpha(\boldsymbol{X}) = \mu\|\boldsymbol{X}\|_{\mathrm{F}}^2$, with $\mu > 0$ is a possible choice.

With a slight abuse of notation, for (2.17) we can write the following subproblem for an arbitrary column $\boldsymbol{c}$ of $\boldsymbol{C}$

$$\min_{\boldsymbol{c}\geq 0} D_{\varphi}(\boldsymbol{Bc};\boldsymbol{a}) + \beta(\boldsymbol{c}). \tag{2.18}$$

Unfortunately, the simple auxiliary function technique of the previous section does not carry over easily for handling (2.18). For example, if $G(\boldsymbol{c},\tilde{\boldsymbol{c}})$ is an auxiliary function for $F(\boldsymbol{c})$, then clearly $G(\boldsymbol{c},\tilde{\boldsymbol{c}}) + \beta(\boldsymbol{c})$ is an auxiliary function for $F(\boldsymbol{c}) + \beta(\boldsymbol{c})$, but even if $G(\boldsymbol{c},\tilde{\boldsymbol{c}})$ is easy to minimize, the nonlinear function $G(\boldsymbol{c},\tilde{\boldsymbol{c}}) + \beta(\boldsymbol{c})$ cannot usually be minimized to obtain a closed form solution. However, when $F(\boldsymbol{c}) = \|\boldsymbol{Bc}-\boldsymbol{a}\|^2$ and $\beta(\boldsymbol{c}) = \frac{\lambda}{2}\|\boldsymbol{c}\|^2$, then $G(\boldsymbol{c},\tilde{\boldsymbol{c}}) + \beta(\boldsymbol{c})$ is a useful auxiliary function, and leads to the (expected) update

$$c_p = \tilde{c}_p\left(\frac{[\boldsymbol{B}^T\boldsymbol{a}]_p}{[\boldsymbol{B}^T\boldsymbol{B}\tilde{\boldsymbol{c}} + \lambda\boldsymbol{I}]_p}\right). \tag{2.19}$$

In general, the problem remains difficult, and it is not easy to minimize $G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) + \beta(\boldsymbol{c})$. Nevertheless, just as in GEM, we could seek to ensure descent, rather than aim for minimization, i.e., we compute $\boldsymbol{c}^{t+1}$ so that

$$G(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t) + \beta(\boldsymbol{c}^{t+1}) \leq G(\boldsymbol{c}^t, \boldsymbol{c}^t) + \beta(\boldsymbol{c}^t). \tag{2.20}$$

A Heuristic. For the sake of simplicity and efficiency, the following heuristic procedure approximates the descent (2.20). While minimizing $G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) + \beta(\boldsymbol{c})$, we approximate $\partial\beta/\partial c_p$ by $\partial\beta/\partial c_p|_{c_p=\tilde{c}_p}$. Then, following the derivation of the previous section, we can obtain for multiplicative $\nabla\varphi$ the following update

$$c_p = \tilde{c}_p \cdot (\nabla\varphi)^{-1}\Big(\frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p - [(\nabla\beta)(\tilde{\boldsymbol{c}})]_p}{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}\Big). \tag{2.21}$$

For update (2.21) we must ensure that the argument of $(\nabla\varphi)^{-1}$ remains within its domain. In practice, this is not a problem.

2.5 Nonlinear Version of (2.4) with "link" functions

Certain nonlinear relationships between the input $\boldsymbol{A}$ and its approximant $\boldsymbol{BC}$ may be modeled by a "link" function that describes the nonlinearity. For example the link function h can be used to model a relation of the form $\boldsymbol{A} \approx h(\boldsymbol{BC})$. Problem (2.4) now becomes

$$\min \quad D_\varphi(h(\boldsymbol{BC}); \boldsymbol{A}), \qquad \boldsymbol{B}, \boldsymbol{C} \geq 0. \tag{2.22}$$

Naturally, solving (2.22) for arbitrary link functions h can be very difficult. However, if the composition $(\varphi \circ h)$ is convex, then we can obtain algorithms

for this problem with link functions without too much difficulty. For simplicity we restrict h to be an elementwise function of its matrix argument.

For example, if h is convex (concave) and φ is an increasing (decreasing) function then, $\varphi \circ h$ is also convex as may be verified by considering the second derivative

$$(\varphi \circ h)''(x) = h''(x)\nabla\varphi(h(x)) + \nabla\varphi'(h(x))(h'(x))^2,$$

which is nonnegative for the specified h and φ. Writing $g = (\varphi \circ h)$ one can verify that

$$\begin{aligned} F(\boldsymbol{c}) = D_\varphi(h(\boldsymbol{B}\boldsymbol{c}); \boldsymbol{a}) &= \sum_i g((\boldsymbol{B}\boldsymbol{c})_i) - \varphi(a_i) - \nabla\varphi(a_i)(h\big((\boldsymbol{B}\boldsymbol{c})_i\big) - a_i) \\ &\leq \sum_{ij} \lambda_{ij} g\left(\frac{b_{ij}c_j}{\lambda_{ij}}\right) - \left(\sum_i \varphi(a_i) + \nabla\varphi(a_i)(h\big((\boldsymbol{B}\boldsymbol{c})_i\big) - a_i)\right). \end{aligned} \tag{2.23}$$

When $\nabla\varphi(x) \geq 0$ (for $x \geq 0$), then using the convexity of h we may also define the divergence

$$0 \leq \sum_i \nabla\varphi(a_i) D_h((\boldsymbol{B}\boldsymbol{c})_i; (\boldsymbol{B}\tilde{\boldsymbol{c}})_i). \tag{2.24}$$

Adding the right hand sides of (2.23) and (2.24) we obtain

$$\begin{aligned} G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = \sum_{ij} \lambda_{ij} g\left(\frac{b_{ij}c_j}{\lambda_{ij}}\right) - \sum_i \varphi(a_i) + \sum_i a_i \nabla\varphi(a_i) - \\ \sum_i \nabla\varphi(a_i)\big\{h((\boldsymbol{B}\tilde{\boldsymbol{c}})_i) + h'((\boldsymbol{B}\tilde{\boldsymbol{c}})_i)((\boldsymbol{B}\boldsymbol{c})_i - (\boldsymbol{B}\tilde{\boldsymbol{c}})_i)\big\}. \end{aligned} \tag{2.25}$$

As in Section 2.3.1, we differentiate the auxiliary function $G(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ and solve

$$\frac{\partial G}{\partial c_p} = \sum_i g'\left(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i\right) b_{ip} - b_{ip} h'((\boldsymbol{B}\tilde{\boldsymbol{c}})_i)\nabla\varphi(a_i) = 0. \tag{2.26}$$

Depending on h, Equation (2.26) may or may not be analytically solvable. If not, then instead of minimizing (2.25) we instead merely do a descent on it (see discussion related to inequality (2.20)).

2.6 Algorithms for Problem (2.5)

We now turn our attention to algorithms for solving Problem (2.5). We present two approaches. Our first approach in Section 2.6.1 below exploits the idea of link functions developed in Section 2.5 above, whereas our second approach (in Section 2.6.2) is based on approximately satisfying the KKT necessary conditions for optimality for the Problem (2.5). The approach with link functions leads to algorithms that are convergent by construction, whereas the approach via KKT conditions we must treat convergence separately.

As before we illustrate our methods for a subproblem of (2.5), namely for

$$\min_{\boldsymbol{c}\geq 0} \quad F(\boldsymbol{c}) = D_{\varphi}(\boldsymbol{a}; \boldsymbol{B}\boldsymbol{c}). \tag{2.27}$$

2.6.1 Solutions via Link Functions.

In this section we exploit our method based upon link functions for providing one class of solutions for (2.5). Link functions arise naturally for Bregman divergences via the following duality relation,

$$D_{\varphi}(x; y) = D_{\varphi^*}(\nabla\varphi(y); \nabla\varphi(x)),$$

where $\varphi^*(x)$ is the *Legendre-conjugate*[1] of φ. This relation allows us to switch the order of arguments to a Bregman divergence. Therefore, whenever $\nabla\varphi$ is convex we can use it as a link function[2] to convert Problem (2.5) into an equivalent problem of the type (2.22). Thus for

$$F(\boldsymbol{c}) = D_{\varphi}(\boldsymbol{a}; \boldsymbol{B}\boldsymbol{c}) = D_{\varphi^*}(\nabla\varphi(\boldsymbol{B}\boldsymbol{c}); \nabla\varphi(\boldsymbol{a})),$$

we follow the derivation in (2.25) and (2.26) with $g = (\varphi^* \circ \nabla\varphi)$, and $\zeta(x) = \nabla\varphi'(x)$ to obtain

$$\frac{\partial G}{\partial c_p} = \sum_i g'(\frac{c_p}{\tilde{c}_p}(\boldsymbol{B}\tilde{\boldsymbol{c}})_i)b_{ip} - b_{ip}\zeta((\boldsymbol{B}\tilde{\boldsymbol{c}})_i)a_i. \tag{2.28}$$

Now using $g'(x) = x\zeta(x)$ we can solve (2.28) to obtain an update for c_p. For e.g., when ζ is multiplicative, (2.28) results in the update

$$c_p\zeta(c_p) = \tilde{c}_p\zeta(\tilde{c}_p)\left(\frac{[\boldsymbol{B}^T\widetilde{\boldsymbol{Z}}\boldsymbol{a}]_p}{[\boldsymbol{B}^T\widetilde{\boldsymbol{Z}}\boldsymbol{B}\tilde{\boldsymbol{c}}]_p}\right), \tag{2.29}$$

where $\widetilde{\boldsymbol{Z}} = \mathrm{diag}(\zeta(\boldsymbol{B}\tilde{\boldsymbol{c}}))$, i.e., $z_{ii} = \zeta((\boldsymbol{B}\tilde{\boldsymbol{c}})_i)$. Despite being somewhat restrictive, this update applies to a wide number of special cases, for e.g., it applies to the problems given in Table 2.1. The update (2.29) guarantees monotonic descent, i.e., $F(\boldsymbol{c}^{t+1}) \leq F(\boldsymbol{c}^t)$, if $\boldsymbol{c}^{t+1} = \boldsymbol{c}$ is computed via (2.29) with $\tilde{\boldsymbol{c}} = \boldsymbol{c}^t$. In Section 2.6.2 below we derive updates for minimizing (2.5) that are more generally applicable than those in this section, and do not depend upon the convexity of $\nabla\varphi$ or separability of ζ (no multiplicativity or additivity assumptions about ζ are made).

[1]The Legendre-conjugate of a convex function is defined as $\varphi^*(y) = \sup_x\big(xy - \varphi(x)\big)$.

[2]Whenever $\nabla\varphi$ is convex, $g = (\varphi^* \circ \nabla\varphi)$ is also convex, since $g''(x) = x\nabla\varphi''(x) + \varphi''(x) \geq 0$, since both $\nabla\varphi''$ and φ'' are non-negative.

2.6.2 Solutions Based on KKT Conditions

The approach in this section is different from the previous sections; the procedures that we derive below are based on approximately solving the KKT necessary conditions to yield multiplicative updates for each component of $\boldsymbol{c}$.

Recall that we need to minimize $F(\boldsymbol{c}) = D_\varphi(\boldsymbol{a}; \boldsymbol{B}\boldsymbol{c})$ subject to $\boldsymbol{c} \geq \boldsymbol{0}$. The Lagrangian is

$$L(\boldsymbol{c}, \boldsymbol{\lambda}) = F(\boldsymbol{c}) - \boldsymbol{\lambda}^T \boldsymbol{c},$$

where $\boldsymbol{\lambda} \geq \boldsymbol{0}$ is the vector of Lagrange multipliers. The KKT necessary conditions for optimality are

$$[\nabla_{\boldsymbol{c}} F(\boldsymbol{c})]_p = \lambda_p \tag{2.30a}$$

$$\lambda_p c_p = 0 \tag{2.30b}$$

$$\lambda_p \geq 0, c_p \geq 0. \tag{2.30c}$$

Letting $\boldsymbol{Z} = \operatorname{diag}(\zeta(\boldsymbol{B}\boldsymbol{c}))$, we combine

$$[\nabla_{\boldsymbol{c}} F(\boldsymbol{c})]_p = [\boldsymbol{B}^T \boldsymbol{Z}(\boldsymbol{B}\boldsymbol{c} - \boldsymbol{a})]_p,$$

with (2.30a) and (2.30b) to obtain

$$[\boldsymbol{B}^T \boldsymbol{Z} \boldsymbol{B}\boldsymbol{c}]_p c_p = [\boldsymbol{B}^T \boldsymbol{Z}\boldsymbol{a}]_p c_p. \tag{2.31}$$

Since $\boldsymbol{c}$ occurs on both sides of (2.31) we solve for it iteratively. Hence, we obtain the following update

$$c_p \leftarrow \tilde{c}_p \frac{[\boldsymbol{B}^T \widetilde{\boldsymbol{Z}} \boldsymbol{a}]_p}{[\boldsymbol{B}^T \widetilde{\boldsymbol{Z}} \boldsymbol{B}\tilde{\boldsymbol{c}}]_p}, \tag{2.32}$$

where $\widetilde{\boldsymbol{Z}} = \text{diag}(\zeta(\boldsymbol{B}\tilde{\boldsymbol{c}}))$. We remark that in practice, we carry out only one iteration of (2.32) instead of iterating it until a fixed point is achieved. As before the overall algorithm can be implemented to run in $O(TKnz(\boldsymbol{A}))$ time. The following lemma shows that the iterative update scheme (2.32) is a reasonable scheme.

Lemma 2.4 (Behavior at Stationarity)**.** *If $\tilde{\boldsymbol{c}}$ is a stationary point of* (2.27) *and we compute $\boldsymbol{c}$ via* (2.32)*, then $\boldsymbol{c} = \tilde{\boldsymbol{c}}$.*

Proof. Let $\tilde{\boldsymbol{c}}$ be a stationary point of (2.27). If $\tilde{c}_p = 0$, then update (2.32) ensures that $c_p = 0$ too. If $\tilde{c}_p \neq 0$, then (2.31), i.e., $[\boldsymbol{B}^T\widetilde{\boldsymbol{Z}}\boldsymbol{a}]_p = [\boldsymbol{B}^T\widetilde{\boldsymbol{Z}}\boldsymbol{B}\tilde{\boldsymbol{c}}]_p$ must hold since $\tilde{\boldsymbol{c}}$ is stationary, whereby update (2.32) yields $c_p = \tilde{c}_p$. □

Now we proceed to analyze what happens when $\tilde{\boldsymbol{c}}$ is not a stationary point. The first important step towards that is to verify that the update (2.32) leads to a monotonic decrease in the objective function value $F(\boldsymbol{c})$. Note that it is not always easy to prove that a monotonic descent procedure converges to a stationary point, without several additional assumptions. For example, in the convergence proof of GEM, a critical assumption is made, namely, that an optimal set of parameter values lies in the interior of the feasible region. For NNMA such an assumption is unrealistic, since in practice many of the entries of both $\boldsymbol{B}$ and $\boldsymbol{C}$ are zero, i.e., they lie at the boundary of the feasible region. Therefore, the best practical approach is to demonstrate monotonic descent under the update (2.32). Even this appears to be very difficult, and Section 2.7 below describes our initial progress towards this goal.

2.7 Monotonicity

To assess the correctness of the update (2.32) we need to show that $F(\boldsymbol{c}) \leq F(\tilde{\boldsymbol{c}})$, i.e., the objective function is non-increasing under the prescribed update. The difference

$$F(\tilde{\boldsymbol{c}}) - F(\boldsymbol{c}) = \sum_i (a_i - (\boldsymbol{B}\boldsymbol{c})_i)(\nabla\varphi((\boldsymbol{B}\boldsymbol{c})_i) - \nabla\varphi((\boldsymbol{B}\tilde{\boldsymbol{c}})_i)) + D_\varphi((\boldsymbol{B}\boldsymbol{c})_i; (\boldsymbol{B}\tilde{\boldsymbol{c}})_i),$$

which in vector notation is

$$F(\tilde{\boldsymbol{c}}) - F(\boldsymbol{c}) = (\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c})^T \big(\nabla\varphi(\boldsymbol{B}\boldsymbol{c}) - \nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})\big) + D_\varphi(\boldsymbol{B}\boldsymbol{c}; \boldsymbol{B}\tilde{\boldsymbol{c}}). \tag{2.33}$$

Let $\delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = F(\tilde{\boldsymbol{c}}) - F(\boldsymbol{c})$; we need to prove that the change $\delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) \geq 0$. It seems difficult to prove this assertion without further assumptions on φ or $\nabla\varphi$. However, for three important cases we show that $\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) - D_\varphi(\boldsymbol{B}\boldsymbol{c}; \boldsymbol{B}\tilde{\boldsymbol{c}}) \geq 0$, which is indeed a stronger statement than just the non-negativity of δ, since D_φ itself is always non-negative.

Lemma 2.5 (Monotonicity)**.** *For $\varphi(x) = \frac{1}{2}x^2$, $\varphi(x) = x\log x$, or $\varphi(x) = -\log x$, if $\boldsymbol{c}$ is obtained by updating $\tilde{\boldsymbol{c}}$ as per* (2.32)*, then*

$$\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) - D_\varphi(\boldsymbol{B}\boldsymbol{c}; \boldsymbol{B}\tilde{\boldsymbol{c}}) \geq 0.$$

Proof. See Sections 2.7.1, 2.7.3, and 2.7.2 below. □

We remark that even though monotonicity for the first two cases ($\varphi(x) = \frac{1}{2}x^2$, and $\varphi(x) = x\log x$) is already known [172], our proofs are *new*. Proving $\hat{\Delta}(\boldsymbol{c}) \geq 0$ directly seems to be very difficult in general. However, intuition behind why we would expect this result to hold in general is provided via a linearization argument in Section 2.7.4.

2.7.1 Least-squares NNMA

For proving monotonicity of the Least-squares NNMA problem we will require the following simple lemma.

Lemma 2.6. *Let $\boldsymbol{X}$ be a symmetric non-negative matrix, and $\boldsymbol{c}, \tilde{\boldsymbol{c}} \geq 0$, then*

$$\sum_i (\boldsymbol{X}\tilde{\boldsymbol{c}})_i \frac{c_i^2}{\tilde{c}_i} - \boldsymbol{c}^T \boldsymbol{X} \boldsymbol{c} \geq 0. \tag{2.34}$$

Proof. Not surprisingly, our proof turns out to be identical to a part of the original proof by Lee and Seung [172], because they also needed to prove a contention similar to (2.34). However, our general approach is direct and we do not require the construction of an auxiliary function. Without loss of generality we may write $c_i = e_i \tilde{c}_i$, and we have

$$\begin{aligned}
S =& \sum_i (\boldsymbol{X}\tilde{\boldsymbol{c}})_i e_i^2 \tilde{c}_i - \sum_{ij} \tilde{c}_i \tilde{c}_j x_{ij} e_i e_j \\
=& \sum_{ij} x_{ij} \tilde{c}_j e_i^2 \tilde{c}_i - \sum_{ij} \tilde{c}_i \tilde{c}_j x_{ij} e_i e_j, \quad \text{and also,} \\
S =& \sum_{ji} x_{ji} \tilde{c}_i e_j^2 \tilde{c}_j - \sum_{ji} \tilde{c}_j \tilde{c}_i x_{ji} e_j e_i \\
=& \sum_{ij} x_{ij} \tilde{c}_i \tilde{c}_j e_j^2 - \sum_{ij} \tilde{c}_i \tilde{c}_j x_{ij} e_i e_j. \quad \text{Therefore,} \\
S =& \frac{1}{2} \sum_{ij} x_{ij} \tilde{c}_i \tilde{c}_j (e_i^2 + e_j^2 - 2 e_i e_j) \\
=& \frac{1}{2} \sum_{ij} x_{ij} \tilde{c}_i \tilde{c}_j (e_i - e_j)^2 \geq 0.
\end{aligned}$$

The third and fourth lines depend upon the symmetry of $\boldsymbol{X}$, and the last step depends upon the non-negativity of $\boldsymbol{X}$ and $\tilde{\boldsymbol{c}}$. □

For the Least-squares NNMA problem the contention of Lemma 2.5 translates into

$$0 \leq \hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \sum_i \big(a_i - (\boldsymbol{B}\boldsymbol{c})_i\big)((\boldsymbol{B}\boldsymbol{c})_i - (\boldsymbol{B}\tilde{\boldsymbol{c}})_i),$$

where $\boldsymbol{c}$ is given by the update (obtained using $\varphi(x) = \frac{1}{2}x^2$ in (2.32))

$$c_i = \tilde{c}_i \frac{(\boldsymbol{B}^T\boldsymbol{a})_i}{(\boldsymbol{B}^T\boldsymbol{B}\tilde{\boldsymbol{c}})_i}. \tag{2.35}$$

Cross-multiplying and summing (2.35) over i we see that

$$\sum_i c_i(\boldsymbol{B}^T\boldsymbol{B}\tilde{\boldsymbol{c}})_i = \sum_i \tilde{c}_i(\boldsymbol{B}^T\boldsymbol{a})_i \Leftrightarrow (\boldsymbol{B}\boldsymbol{c})^T(\boldsymbol{B}\tilde{\boldsymbol{c}}) = \boldsymbol{a}^T(\boldsymbol{B}\tilde{\boldsymbol{c}}), \tag{2.36}$$

whereby $\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \boldsymbol{c}^T\boldsymbol{B}^T\boldsymbol{a} - \boldsymbol{c}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{c}$. Using (2.35) we can eliminate $(\boldsymbol{B}^T\boldsymbol{a})_i$ to obtain

$$\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \sum_i \frac{c_i^2}{\tilde{c}_i}(\boldsymbol{B}^T\boldsymbol{B}\tilde{\boldsymbol{c}})_i - c_i(\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{c})_i.$$

Now, using Lemma 2.6 with $\boldsymbol{X} = \boldsymbol{B}^T\boldsymbol{B}$ we immediately obtain $\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) \geq 0$.

2.7.2 KL-Divergence NNMA

For the KL-Divergence NNMA problem we have $\nabla\varphi(x) = 1 + \log x$. Thus, the contention of Lemma 2.6 becomes need to prove that

$$0 \leq \hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \sum_i (a_i - (\boldsymbol{B}\boldsymbol{c})_i) \log \frac{(\boldsymbol{B}\boldsymbol{c})_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}.$$

We proceed by analyzing individual terms in the summation above. We have

$$\begin{aligned}\log\frac{(\boldsymbol{Bc})_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} &= \frac{1}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}\Big((\boldsymbol{B}\tilde{\boldsymbol{c}})_i \log\frac{(\boldsymbol{Bc})_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}\Big) \\ &= \frac{1}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}\Big(\sum_j b_{ij}\tilde{c}_j \log\frac{\sum_l b_{il}c_l}{\sum_l b_{il}\tilde{c}_l}\Big) \\ &\geq \frac{1}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}\sum_j b_{ij}\tilde{c}_j \log\frac{c_j}{\tilde{c}_j}, \end{aligned} \tag{2.37}$$

where the latter inequality follows from the log-sum inequality

$$\sum_i x_i \log\frac{x_i}{y_i} \geq (\sum_i x_i)\log\frac{\sum_i x_i}{\sum_i y_i}.$$

A second application of this log-sum inequality allows us to conclude that

$$-(\boldsymbol{Bc})_i \log\frac{(\boldsymbol{Bc})_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} \geq -\sum_j b_{ij}c_j \log\frac{c_j}{\tilde{c}_j}. \tag{2.38}$$

Adding (2.37) and (2.38) we have

$$\begin{aligned}\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) &\geq \sum_{ij} b_{ij}\log\frac{c_j}{\tilde{c}_j}\Big(\frac{a_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}\tilde{c}_j - c_j\Big) \\ &= \sum_j \log\frac{c_j}{\tilde{c}_j}\Big((\tilde{c}_j\sum_i b_{ij}\frac{a_i}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}) - \sum_i b_{ij}c_j\Big) \\ &= 0, \end{aligned}$$

where the last equality follows from the update (2.32), which says

$$c_j = \tilde{c}_j\frac{\sum_i b_{ij}a_i/(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}{\sum_i b_{ij}}.$$

Hence the proof is complete.

2.7.3 Burg-Entropy NNMA

When $\varphi(x) = -\log x$ ($\operatorname{dom}\varphi$ is $\mathbb{R}_{++}$), we obtain the Burg-Entropy NNMA problem. The contention of Lemma 2.5 then becomes

$$0 \leq \hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) = \sum_i \big(a_i - (\boldsymbol{B}\boldsymbol{c})_i\big)\left(\frac{1}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} - \frac{1}{(\boldsymbol{B}\boldsymbol{c})_i}\right).$$

For $\varphi(x) = -\log x$, $\zeta(x) = \varphi''(x) = 1/x^2$, so that the update (2.32) yields

$$c_i = \tilde{c}_i \frac{(\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{a})_i}{(\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{B}\tilde{\boldsymbol{c}})_i}, \tag{2.39}$$

where $\boldsymbol{Z} = \operatorname{diag}(1/(\boldsymbol{B}\tilde{\boldsymbol{c}})_i^2)$. Cross-multiplying and summing (2.39) over i we obtain

$$\sum_{ij} c_i b_{ij}/(\boldsymbol{B}\tilde{\boldsymbol{c}})_j = \sum_{ij} \tilde{c}_i b_{ij} a_j/(\boldsymbol{B}\tilde{\boldsymbol{c}})_j^2 \quad \Longleftrightarrow \quad \sum_j \frac{(\boldsymbol{B}\boldsymbol{c})_j}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} = \sum_j \frac{a_j}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}.$$

Hence proving $\hat{\delta}(\tilde{\boldsymbol{c}}, \boldsymbol{c}) \geq 0$ boils down to proving

$$\sum_i \left(1 - \frac{a_i}{(\boldsymbol{B}\boldsymbol{c})_i}\right) \geq 0.$$

Applying convexity of $1/x$ to $1/(\boldsymbol{B}\boldsymbol{c})_i$ we obtain

$$\frac{a_i}{(\boldsymbol{B}\boldsymbol{c})_i} = \frac{a_i}{\sum_j b_{ij}c_j} \leq a_i \sum_j \frac{\lambda_{ij}}{b_{ij}c_j/\lambda_{ij}},$$

where $\sum_j \lambda_{ij} = 1$. Letting $\lambda_{ij} = b_{ij}\tilde{c}_j/(\boldsymbol{B}\tilde{\boldsymbol{c}})_i$ we have

$$\begin{aligned}
&\sum_i \frac{a_i}{(\boldsymbol{B}\boldsymbol{c})_i} && \leq && \sum_{ij} \frac{a_i b_{ij} \tilde{c}_j^2}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i^2 c_j} \\
=&\sum_{ij} \frac{a_i b_{ij} \tilde{c}_j (\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{B}\tilde{\boldsymbol{c}})_j}{(\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{a})_j (\boldsymbol{B}\tilde{\boldsymbol{c}})_i^2} && = && \sum_j \frac{(\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{B}\tilde{\boldsymbol{c}})_j \tilde{c}_j}{(\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{a})_j} \sum_i b_{ij} \frac{1}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i^2} a_i \\
=&\sum_j (\boldsymbol{B}^T\boldsymbol{Z}\boldsymbol{B}\tilde{\boldsymbol{c}})_j \tilde{c}_j && = && \sum_{ji} b_{ij} (\boldsymbol{Z}\boldsymbol{B}\tilde{\boldsymbol{c}})_i \tilde{c}_j \\
=&\sum_{ji} \frac{b_{ij}c_j}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} && = && \sum_i 1,
\end{aligned}$$

thereby completing the proof.

2.7.4 Monotonicity with Linearization

In this section we prove that when $\boldsymbol{Bc}$ is close to $\boldsymbol{B\tilde{c}}$, then the update (2.32) guarantees monotonic descent. The main purpose of this section is to provide some intuition behind why we expect Lemma 2.5 to hold for a large class of convex functions φ. It is of course, unreasonable to expect that Lemma to hold for all convex functions with domain $\mathbb{R}_+$ without further qualifications.

First we introduce a lemma that is merely an application of Lemma 2.6, and will be useful for our proof based on linearization.

Lemma 2.7 (Monotonicity)**.** *If $\boldsymbol{c}$ is computed via the update* (2.32)*, then*

$$(\boldsymbol{a} - \boldsymbol{Bc})^T \widehat{\boldsymbol{Z}}(\boldsymbol{Bc} - \boldsymbol{B\tilde{c}}) \geq 0, \tag{2.40}$$

where $\widehat{\boldsymbol{Z}} = \operatorname{diag}(\zeta(\boldsymbol{B\tilde{c}}))$ as before.

Proof. From (2.32) we have

$$c_i[\boldsymbol{B}^T\widehat{\boldsymbol{Z}}\boldsymbol{a}]_i = [\boldsymbol{B}^T\widehat{\boldsymbol{Z}}\boldsymbol{B}\tilde{\boldsymbol{c}}]_i\frac{c_i^2}{\tilde{c}_i} \quad \Longrightarrow \quad \boldsymbol{a}^T\widehat{\boldsymbol{Z}}\boldsymbol{Bc} = \sum_i [\boldsymbol{B}^T\widehat{\boldsymbol{Z}}\boldsymbol{B}\tilde{\boldsymbol{c}}]_i\frac{c_i^2}{\tilde{c}_i}.$$

Similarly (2.32) also yields $(\boldsymbol{Bc})^T\widehat{\boldsymbol{Z}}\boldsymbol{B\tilde{c}} = \boldsymbol{a}^T\widehat{\boldsymbol{Z}}\boldsymbol{B\tilde{c}}$. Thus the claim (2.40) reduces to

$$\sum_i [\boldsymbol{B}^T\widehat{\boldsymbol{Z}}\boldsymbol{B}\tilde{\boldsymbol{c}}]_i\frac{c_i^2}{\tilde{c}_i} - (\boldsymbol{Bc})^T\widehat{\boldsymbol{Z}}\boldsymbol{Bc} \geq 0.$$

Now we simply invoke Lemma 2.6 with $\boldsymbol{X} = \boldsymbol{B}^T\widehat{\boldsymbol{Z}}\boldsymbol{B}$ to conclude the truth of this inequality. □

Next we discuss monotonicity by considering the first order Taylor-series expansion

$$\nabla\varphi((\boldsymbol{Bc})_i) \approx \nabla\varphi((\boldsymbol{B\tilde{c}})_i) + \zeta((\boldsymbol{B\tilde{c}})_i)((\boldsymbol{Bc})_i - (\boldsymbol{B\tilde{c}})_i), \tag{2.41}$$

where we neglect the higher-order terms. Thus, using the linearization (2.41) we can write

$$\delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) \approx (\boldsymbol{a} - \boldsymbol{Bc})^T \widehat{\boldsymbol{Z}}(\boldsymbol{Bc} - \boldsymbol{B\tilde{c}}) + D_\varphi(\boldsymbol{Bc}; \boldsymbol{B\tilde{c}}).$$

From Lemma 2.7 we know that $(\boldsymbol{a}-\boldsymbol{Bc})^T \widehat{\boldsymbol{Z}}(\boldsymbol{Bc}-\boldsymbol{B\tilde{c}}) \geq 0$, while $D_\varphi(\boldsymbol{Bc}; \boldsymbol{B\tilde{c}}) \geq 0$ by definition. Thus, we immediately conclude $\delta(\tilde{\boldsymbol{c}}, \boldsymbol{c}) \geq 0$.

2.8 Regularized Version of Problem (2.5)

We now look at the case with nonzero (differentiable) penalty functions by proceeding along the same lines as Section 2.6.2. Other authors have also obtained similar updates with penalty functions, e.g., [50, 51].

Problem (2.5) now takes the form

$$\min_{\boldsymbol{B},\boldsymbol{C}\geq 0} \quad D_\varphi(\boldsymbol{A}; \boldsymbol{BC}) + \alpha(\boldsymbol{B}) + \beta(\boldsymbol{C}), \tag{2.42}$$

where α and β are regularizing functions as discussed previously in Section 2.4. As before, we consider the subproblem

$$\min_{\boldsymbol{c}\geq 0} \quad F(\boldsymbol{c}) = D_\varphi(\boldsymbol{a}; \boldsymbol{Bc}) + \beta(\boldsymbol{c}), \tag{2.43}$$

for which we form the Lagrangian, differentiate it, and obtain the KKT necessary conditions for optimality

$$\nabla_{\boldsymbol{c}}[F(\boldsymbol{c})] = \boldsymbol{\lambda} \tag{2.44a}$$

$$\lambda_p c_p = 0 \tag{2.44b}$$

$$\lambda_p \geq 0, c_p \geq 0. \tag{2.44c}$$

We use (2.44a) and (2.44b), in conjunction with the gradient

$$\nabla_{\boldsymbol{c}}(F(\boldsymbol{c})) = \boldsymbol{B}^T \boldsymbol{Z}(\boldsymbol{B}\boldsymbol{c} - \boldsymbol{a}) + \nabla_{\boldsymbol{c}}\beta(\boldsymbol{c}),$$

to obtain the iterative update

$$c_p \leftarrow \tilde{c}_p \frac{[\boldsymbol{B}^T \widetilde{\boldsymbol{Z}} \boldsymbol{a}]_p}{[\boldsymbol{B}^T \widetilde{\boldsymbol{Z}} \boldsymbol{B} \tilde{\boldsymbol{c}}]_p + [\nabla_{\boldsymbol{c}}\beta(\boldsymbol{c})]_p}. \tag{2.45}$$

The update for $\boldsymbol{b}$ may be derived similarly and we skip it for brevity. If $[\nabla_{\boldsymbol{c}}\beta(\boldsymbol{c})]_p \geq 0$ we do not have to do any additional work to enforce nonnegativity of c_p. Using a non-decreasing penalty function $\beta(\boldsymbol{c})$ can be useful, as the following lemma shows.

Lemma 2.8 (Affect of the Penalty)**.** *If β is an increasing function and $\beta(\tilde{\boldsymbol{c}}) - \beta(\boldsymbol{c}) \geq 0$, then an iterative application of* (2.45) *leads to a fixed point (of the update, and hence of $F(\boldsymbol{c})$).*

Proof. Since β is increasing the hypothesis $\beta(\tilde{\boldsymbol{c}}) - \beta(\boldsymbol{c}) \geq 0$ implies that $\|\tilde{\boldsymbol{c}}\| \geq \|\boldsymbol{c}\|$, whence the update (2.45) is non-expansive, and by Brouwer's fixed point theorem we can conclude that it has a fixed point. □

Remark: The assumption $\beta(\tilde{\boldsymbol{c}}) - \beta(\boldsymbol{c}) \geq 0$ seems to be quite stringent, and in general it can be difficult to verify for arbitrary β. However, it is not difficult to select appropriate β that satisfy this restriction. Note that every fixed point of (2.45) satisfies this inequality with equality.

2.9 Examples of NNMA Problems

In this section we present some specific NNMA problems and their solutions as obtained by an application of our generic approach described above. Later, in Section 2.11.2 we review several of the known applications of NNMA to highlight its wide applicability, and to suggest possible domains where our new algorithms could be applied. In addition, via some examples below, we also derive additional generalizations such as weighted and multi-factor NNMA problems, which also can have many practical uses. These are not given separate treatment because their derivation follows by a straightforward generalization of our methods of §§2.3, 2.6. We distinguish our (new) contributions from examples of previously existing algorithms by suffixing a $\star$ suffixed to the section name. Already existing algorithms are shown to highlight how they turn out to be special cases of our general approach.

2.9.1 New KL-Divergence NNMA*

The original NNMA problem [174] focused on minimizing $\mathrm{KL}(\boldsymbol{a};\boldsymbol{Bc})$. We look at the corresponding asymmetric case that minimizes

$$\mathrm{KL}(\boldsymbol{Bc},\boldsymbol{a}) \;=\; \sum_i (\boldsymbol{Bc})_i \log \frac{(\boldsymbol{Bc})_i}{a_i} - (\boldsymbol{Bc})_i + a_i, \qquad \boldsymbol{B},\boldsymbol{c} \geq 0. \tag{2.46}$$

Let $\varphi(x) = x\log x - x$. Then, $\nabla\varphi(x) = \log x$, and since $\nabla\varphi(xy) = \nabla\varphi(x) + \nabla\varphi(y)$, upon substituting for $\nabla\varphi$ in (2.10) and setting the resultant to zero we obtain

$$\begin{aligned}
&\frac{\partial G}{\partial c_p} = \sum_i b_{ip}\log(c_p(\boldsymbol{B}\tilde{\boldsymbol{c}})_i/\tilde{c}_p) - \sum_i b_{ip}\log a_i = 0,\\
&\implies (\boldsymbol{B}^T\mathbf{1})_p \log\frac{c_p}{\tilde{c}_p} = [\boldsymbol{B}^T\log\boldsymbol{a} - \boldsymbol{B}^T\log(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p\\
&\implies c_p = \tilde{c}_p\cdot\exp\left(\frac{[\boldsymbol{B}^T\log\big(\boldsymbol{a}/(\boldsymbol{B}\tilde{\boldsymbol{c}})\big)]_p}{[\boldsymbol{B}^T\mathbf{1}]_p}\right).
\end{aligned}$$

Similarly the update for $\boldsymbol{b}$ is derived to be

$$b_p = \tilde{b}_p\cdot\exp\left(\frac{[\big(\log(\boldsymbol{a}/\tilde{\boldsymbol{b}}^T\boldsymbol{C})\big)^T\boldsymbol{C}^T]_p}{[\mathbf{1}^T\boldsymbol{C}^T]_p}\right).$$

Due to the $\exp(\cdot)$ function it is obvious that the updates maintain the non-negativity of c_p and b_p once started with non-negative initial values.

2.9.2 Regularized KL-Divergence*

Consider the regularized NNMA sub-problem

$$\min_{\boldsymbol{c}\geq 0}\ \mathrm{KL}(\boldsymbol{Bc},\boldsymbol{a}) \;=\; \sum_i (\boldsymbol{Bc})_i \log \frac{(\boldsymbol{Bc})_i}{a_i} - (\boldsymbol{Bc})_i + a_i + \mu\|\boldsymbol{c}\|_1, \tag{2.47}$$

where we have $\mu > 0$ is a regularization parameter. Note that since $\boldsymbol{c} \geq 0$, we have $\mu\|\boldsymbol{c}\|_1 = \mu \mathbf{1}^T \boldsymbol{c}$. Thus, following the derivation of §2.9.1 we obtain the update

$$c_p = \tilde{c}_p \cdot \exp\left(\frac{[\boldsymbol{B}^T \log\big(\boldsymbol{a}/(\boldsymbol{B}\tilde{\boldsymbol{c}})\big)]_p - \mu}{[\boldsymbol{B}^T \mathbf{1}]_p} \right). \tag{2.48}$$

2.9.3 Original KL-Divergence NNMA and Regularized Versions*

Since the KL-Divergence is convex in both arguments, we can use our auxiliary function technique also for the traditional KL-Divergence NNMA subproblem

$$\mathrm{KL}(\boldsymbol{a}\|\boldsymbol{B}\boldsymbol{c}) = \sum_i a_i \log \frac{a_i}{(\boldsymbol{B}\boldsymbol{c})_i} - a_i + (\boldsymbol{B}\boldsymbol{c})_i.$$

Here we exploit the convexity of $-\log x$ (applied to $x = (\boldsymbol{B}\boldsymbol{c})_i$) to obtain the auxiliary function

$$G(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = \sum_i a_i \log a_i - a_i + (\boldsymbol{B}\boldsymbol{c})_i - \sum_{ij} \frac{a_i b_{ij} \tilde{c}_j}{(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} \log \frac{c_j (\boldsymbol{B}\tilde{\boldsymbol{c}})_i}{\tilde{c}_j}.$$

Now, differentiating G w.r.t. c_p as before and solving $\partial G / \partial c_p = 0$ we obtain exactly Lee & Seung's algorithm for KL-Divergence NNMA. Our derivation is simpler and more direct. Furthermore, our auxiliary function technique immediately yields algorithms for the following regularized KL-Divergence problems:

$$\min_{\boldsymbol{c} \geq 0} \ \mathrm{KL}(\boldsymbol{a}\|\boldsymbol{B}\boldsymbol{c}) = \sum_i a_i \log \frac{a_i}{(\boldsymbol{B}\boldsymbol{c})_i} - a_i + (\boldsymbol{B}\boldsymbol{c})_i + \beta(\boldsymbol{c}), \tag{2.49}$$

where $\beta(\boldsymbol{c}) = \mu\|\boldsymbol{c}\|_1$, or $\frac{\mu}{2}\|\boldsymbol{c}\|_2^2$. The respective updates for the different choices of $\beta(\boldsymbol{c})$ are

$$c_p = \tilde{c}_p \frac{\sum_i a_i b_{ip}/(\boldsymbol{B}\tilde{\boldsymbol{c}})_i}{\mu + \sum_i b_{ip}}, \tag{2.50}$$

$$c_p = \frac{\tilde{c}_p}{2\mu}\left(\sqrt{\tilde{c}_p^{-2}\big(\textstyle\sum_i b_{ip}\big)^2 + 4\mu\tilde{c}_p^{-1}\sum_i a_i b_{ip}/(\boldsymbol{B}\tilde{\boldsymbol{c}})_i} - \textstyle\sum_i b_{ip}\right). \tag{2.51}$$

Observe that the non-negativity of c_p is easily ensured by the updates (2.50) and (2.51).

2.9.4 Constrained NNMA and Maximum Entropy*

Consider the following NNMA subproblem with additional linear inequality constraints

$$\begin{aligned} \min_{\boldsymbol{c}\geq 0} \quad & D_\varphi(\boldsymbol{B}\boldsymbol{c};\boldsymbol{a}) \\ \text{subject to} \quad & \boldsymbol{P}\boldsymbol{c} \leq \boldsymbol{0}. \end{aligned} \tag{2.52}$$

An easy way to tackle this problem is to introduce a differentiable penalty function for enforcing the constraints $\boldsymbol{P}\boldsymbol{c} \leq \boldsymbol{0}$. Thus,

$$F(\boldsymbol{c}) = D_\varphi(\boldsymbol{B}\boldsymbol{c};\boldsymbol{a}) + \rho\|\max(0, \boldsymbol{P}\boldsymbol{c})\|^2, \tag{2.53}$$

where $\rho > 0$ is some penalty parameter. Assuming multiplicative $\nabla\varphi$ and following the auxiliary function technique described in Section 2.4, we obtain the following updates for $\boldsymbol{c}$,

$$c_p \leftarrow \tilde{c}_p \cdot (\nabla\varphi)^{-1}\left(\frac{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{a})]_p - \rho[\boldsymbol{P}^T(\boldsymbol{P}\tilde{\boldsymbol{c}})^+]_p}{[\boldsymbol{B}^T\nabla\varphi(\boldsymbol{B}\tilde{\boldsymbol{c}})]_p}\right),$$

where $(\boldsymbol{P}\boldsymbol{c})^+ = \max(\boldsymbol{0}, \boldsymbol{P}\boldsymbol{c})$. Note that care must be taken to ensure that the addition of the penalty term does not violate the nonnegativity of $\boldsymbol{c}$, and that the argument of $(\nabla\varphi)^{-1}$ lies in its domain.

Maximum Entropy. Incorporating additional constraints into (2.46) is easier, since the exponential updates ensure non-negativity. When $\boldsymbol{a} = \mathbf{1}$, under additional linear inequality constraints, Problem (2.46) becomes

$$\min_{\boldsymbol{c}\geq 0} \quad \mathrm{KL}(\boldsymbol{B}\boldsymbol{c}\|\mathbf{1}) \qquad \text{s.t. } \boldsymbol{P}\boldsymbol{c} \leq 0.$$

Using the penalty function method as for (2.53) we obtain

$$c_p \quad \leftarrow \quad c_p \cdot \exp\left(\frac{[-\boldsymbol{B}^T \log(\boldsymbol{B}\boldsymbol{c}) - \rho \boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{c})^+]_p}{[\boldsymbol{B}^T\mathbf{1}]_p}.\right)$$

Note that one may use other penalty functions to enforce $\boldsymbol{P}\boldsymbol{c} \leq \mathbf{0}$, provided that these functions are differentiable.

2.9.5 Lee and Seung's Algorithms.

When $\varphi(x) = \frac{1}{2}x^2$ our KKT technique of § 2.6.2 (Update (2.32)) yields

$$b_{mk} \leftarrow b_{mk}\frac{(\boldsymbol{A}\boldsymbol{C}^T)_{mk}}{(\boldsymbol{B}\boldsymbol{C}\boldsymbol{C}^T)_{mk}}, \qquad c_{kn} \leftarrow c_{kn}\frac{(\boldsymbol{B}^T\boldsymbol{A})_{kn}}{(\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C})_{kn}},$$

while for $\varphi(x) = x\log x$ we obtain

$$b_{mk} \leftarrow b_{mk}\left\{\frac{\left([\frac{\boldsymbol{A}}{\boldsymbol{B}\boldsymbol{C}}]\boldsymbol{C}^T\right)_{mk}}{(\mathbf{1}_M\mathbf{1}_N^T\boldsymbol{C}^T)_{mk}} \;=\; \frac{\sum_s c_{ks}a_{ms}/(\boldsymbol{B}\boldsymbol{C})_{ms}}{\sum_n c_{kn}}\right\},$$
$$c_{kn} \leftarrow c_{kn}\left\{\frac{\left(\boldsymbol{B}^T[\frac{\boldsymbol{A}}{\boldsymbol{B}\boldsymbol{C}}]\right)_{kn}}{(\boldsymbol{B}^T\mathbf{1}_M\mathbf{1}_N^T)_{kn}} \;=\; \frac{\sum_t b_{tk}a_{tn}/(\boldsymbol{B}\boldsymbol{C})_{tn}}{\sum_m b_{mk}}\right\}.$$

These updates are the same as the ones originally derived by Lee and Seung [172]. It can be shown that these updates are essentially equivalent (see [102]) to those for probabilistic latent semantic indexing (PLSI) [130], though due to the extra normalization in PLSI a subtle difference remains. In particular, a

fixed point of NNMA yields a fixed point of PLSI and vice-versa, but beginning from the same initialization, both usually end up reaching different final solutions.

2.9.6 The Multi-factor NNMA Problem*

Both the NNMA problems (2.4) and (2.5) can be extended to the "multi-factor" problem, wherein one seeks an approximation of the type $\boldsymbol{A} \approx \boldsymbol{B}_1\boldsymbol{B}_2 \ldots \boldsymbol{B}_R$. As a simple example, consider minimizing

$$D_\varphi(\boldsymbol{A}; \boldsymbol{B}_1\boldsymbol{B}_2 \ldots \boldsymbol{B}_R),$$

where all matrices involved are nonnegative. We compute the gradient of the divergence D_φ w.r.t. each $\boldsymbol{B}_r$. Let $\widehat{\boldsymbol{B}} = \boldsymbol{B}_1\boldsymbol{B}_2 \ldots \boldsymbol{B}_{r-1}$, $\widehat{\boldsymbol{C}} = \boldsymbol{B}_{r+1}\boldsymbol{B}_{r+2} \ldots \boldsymbol{B}_R$, and $\boldsymbol{H} = \boldsymbol{B}_1\boldsymbol{B}_2 \ldots \boldsymbol{B}_R$. Let b^r_{pq} denote $(\boldsymbol{B}_r)_{pq}$. It is easy to verify that

$$\frac{\partial D_\varphi}{\partial b^r_{pq}} = \left[\widehat{\boldsymbol{B}}^T(\zeta(\boldsymbol{H}) \odot (\boldsymbol{H} - \boldsymbol{A}))\widehat{\boldsymbol{C}}^T\right]_{pq}.$$

Following the derivation in Section 2.6.2 we obtain the update

$$\boldsymbol{B}_r \leftarrow \boldsymbol{B}_r \odot \frac{\widehat{\boldsymbol{B}}^T(\zeta(\boldsymbol{H}) \odot \boldsymbol{A})\widehat{\boldsymbol{C}}^T}{\widehat{\boldsymbol{B}}^T(\zeta(\boldsymbol{H}) \odot \boldsymbol{H})\widehat{\boldsymbol{C}}^T}. \tag{2.54}$$

Below we show a simple application of (2.54) for the special case of $R = 3$.

Lemma 2.9 (Multi-factor monotonicity)**.** *For $\varphi(x) = \frac{1}{2}x^2, x\log x$, and $-\log x$, the multifactor update* (2.54) *ensures monotonic descent in the objective function $D_\varphi(\boldsymbol{A}; \boldsymbol{B}_1\boldsymbol{B}_2 \ldots \boldsymbol{B}_R)$.*

Proof. Easy generalization by induction of the proof of Lemma 2.5. □

2.9.6.1 Application to Relaxed Co-clustering⋆

A simple example of multi-factor NNMA is to the three-factor NNMA, namely $\boldsymbol{A} \approx \boldsymbol{RBC}^T$. Such an approximation is closely tied to the problem of co-clustering [49], and also provides a richer model than PLSI [130]. The matrices $\boldsymbol{R}$, $\boldsymbol{C}$, $\boldsymbol{B}$ may be thought of as representing row clusters, column clusters, and co-cluster prototypes, respectively. For example, for the problems (all matrices are non-negative)

$$\begin{aligned} &\min \ \ \|\boldsymbol{A} - \boldsymbol{RBC}^T\|_{\mathrm{F}}^2 && \text{(Euclidean)} \\ &\min \ \ \mathrm{KL}(\boldsymbol{A} \| \boldsymbol{RBC}^T) && \text{(Information-theoretic)}, \end{aligned}$$

an application of (2.54) yields the updates

$$\boldsymbol{R} \odot \frac{\boldsymbol{ACB}^T}{\boldsymbol{RBC}^T\boldsymbol{CB}^T}, \quad \boldsymbol{B} \odot \frac{\boldsymbol{R}^T\boldsymbol{AC}}{\boldsymbol{R}^T\boldsymbol{RBC}^T\boldsymbol{C}}, \quad \boldsymbol{C} \odot \frac{\boldsymbol{A}^T\boldsymbol{RB}}{\boldsymbol{CB}^T\boldsymbol{R}^T\boldsymbol{RB}}, \quad (E)$$

$$\boldsymbol{R} \odot \frac{[\frac{\boldsymbol{A}}{\boldsymbol{RBC}^T}]\boldsymbol{CB}^T}{\mathbf{1}\mathbf{1}^T\boldsymbol{CB}^T}, \quad \boldsymbol{B} \odot \frac{\boldsymbol{R}^T[\frac{\boldsymbol{A}}{\boldsymbol{RBC}^T}]\boldsymbol{C}}{\boldsymbol{R}^T\mathbf{1}\mathbf{1}^T\boldsymbol{C}}, \quad \boldsymbol{C} \odot \frac{[\frac{\boldsymbol{A}^T}{\boldsymbol{CB}^T\boldsymbol{R}^T}]\boldsymbol{RB}}{\mathbf{1}\mathbf{1}^T\boldsymbol{RB}}, \quad (IT)$$

for the relaxations of the Euclidean and Information-theoretic clustering, respectively. To ensure uniqueness, we normalize both $\boldsymbol{R}$ and $\boldsymbol{C}$ while suitably adjusting the matrix $\boldsymbol{B}$. It is evident that we can exploit the generality of the update (2.54) to obtain relaxed co-clustering solutions to problems that seek to minimize $D_\varphi(\boldsymbol{A}; \boldsymbol{RBC}^T)$. We remark that in practice, the above updates should be implemented to exploit the sparsity of $\boldsymbol{A}$.

2.9.7 Weighted NNMA Problems⋆

There are three main ways in which weighting may be incorporated into the NNMA model. First, we weight the objective function elementwise, which

is useful when one wants to ascribe different weights to different subproblems. Second, we can add elementwise weighting to the low-rank approximation, a flexibility very useful for dealing with measurement uncertainties and missing values in $\boldsymbol{A}$. Finally, we also permit weighted scaling of the low-rank approximation, which is closer to the traditional diagonal weighting where one weights different rows (objects) or columns differently while constructing an approximation. In summary, these three choices, lead to the following six different weighted NNMA problems:

$$\begin{array}{llll} \min \sum_{ij} w_{ij} D_\varphi(a_{ij}; (\boldsymbol{BC})_{ij}) & \text{and} & \min \sum_{ij} w_{ij} D_\varphi((\boldsymbol{BC})_{ij}; a_{ij}), \\ \min \quad D_\varphi(\boldsymbol{A}; \boldsymbol{W} \odot (\boldsymbol{BC})) & \text{and} & \min \quad D_\varphi(\boldsymbol{W} \odot (\boldsymbol{BC}); \boldsymbol{A}), \\ \min \quad D_\varphi(\boldsymbol{A}; \boldsymbol{W}_1 \boldsymbol{BC} \boldsymbol{W}_2) & \text{and} & \min \quad D_\varphi(\boldsymbol{W}_1 \boldsymbol{BC} \boldsymbol{W}_2; \boldsymbol{A}), \end{array}$$

where we assume the weighting matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ to be themselves nonnegative. All of these problems may be solved easily by the auxiliary function and KKT based techniques that we have developed above. To avoid unnecessary repetition we skip the derivations of the associated updates.

The PMF Problem. We seek to minimize $\frac{1}{2}\|\sqrt{\boldsymbol{W}} \odot (\boldsymbol{A} - \boldsymbol{BC})\|_{\mathrm{F}}^2$. Following the techniques for deriving (2.32) we obtain the updates

$$\boldsymbol{B} \leftarrow \boldsymbol{B} \odot \frac{(\boldsymbol{W} \odot \boldsymbol{A})\boldsymbol{C}^T}{(\boldsymbol{W} \odot (\boldsymbol{BC}))\boldsymbol{C}^T}, \qquad \boldsymbol{C} \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{B}^T(\boldsymbol{W} \odot \boldsymbol{A})}{\boldsymbol{B}^T(\boldsymbol{W} \odot (\boldsymbol{BC}))}. \tag{2.55}$$

These iterative updates are significantly simpler than the PMF algorithms of [214] and may be used as alternatives to them.

Lemma 2.10 (Monotonicity for PMF)**.** *The elementwise weighted updates* (2.55) *monotonically decrease their corresponding objective function.*

Proof. Notice that for a single column $\boldsymbol{c}$, the difference in objective function value after the update (2.55) is given by

$$\sum_i w_i \Big((a_i - (\boldsymbol{B}\boldsymbol{c})_i)((\boldsymbol{B}\boldsymbol{c})_i - (\boldsymbol{B}\tilde{\boldsymbol{c}})_i) + \tfrac{1}{2}\big((\boldsymbol{B}\boldsymbol{c})_i - (\boldsymbol{B}\tilde{\boldsymbol{c}})_i\big)^2 \Big).$$

Therefore we can mimic the proof in §2.7.1 to conclude this lemma. □

Application to Weighted KL-Divergence. Here we wish to minimize $\mathrm{KL}(\boldsymbol{A}; \boldsymbol{P}\boldsymbol{B}\boldsymbol{C}\boldsymbol{Q})$, where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are positive diagonal matrices. This problem is a slight generalization of the diagonally weighted problem considered by [115, 116]. Using (2.54), we obtain

$$b_{mk} \leftarrow b_{mk} \left\{ \frac{\big(\boldsymbol{P}\big[\frac{\boldsymbol{A}}{\boldsymbol{P}\boldsymbol{B}\boldsymbol{C}\boldsymbol{Q}}\big]\boldsymbol{Q}\boldsymbol{C}^T\big)_{mk}}{\big(\boldsymbol{P}(\mathbf{1}_M\mathbf{1}_N^T)\boldsymbol{Q}\boldsymbol{C}^T\big)_{mk}} \quad = \quad \frac{\sum_s c_{ks}a_{ms}/(p_{mm}(\boldsymbol{B}\boldsymbol{C})_{ms})}{\sum_n c_{kn}q_{nn}} \right\} \tag{2.56}$$

$$c_{kn} \leftarrow c_{kn} \left\{ \frac{\big(\boldsymbol{B}^T\boldsymbol{P}\big[\frac{\boldsymbol{A}}{\boldsymbol{P}\boldsymbol{B}\boldsymbol{C}\boldsymbol{Q}}\big]\boldsymbol{Q}\big)_{kn}}{\big(\boldsymbol{B}^T\boldsymbol{P}(\mathbf{1}_M\mathbf{1}_N^T)\boldsymbol{Q}\big)_{kn}} \quad = \quad \frac{\sum_t b_{tk}a_{tn}/(q_{nn}(\boldsymbol{B}\boldsymbol{C})_{tn})}{\sum_m b_{mk}p_{mm}} \right\}. \tag{2.57}$$

Observe that when $\boldsymbol{P} = \boldsymbol{I}_M$ and $\boldsymbol{Q} = \boldsymbol{I}_N$ then these updates simplify to those given in Section 2.9.5.

Lemma 2.11 (Monotonicity for Weighted-KL)**.** *The updates* (2.56) *and* (2.57) *monotonically decrease the objective function, i.e., KL*$(\boldsymbol{A}\|\boldsymbol{P}\boldsymbol{B}\boldsymbol{C}\boldsymbol{Q})$*.*

Proof. Immediate generalization of the proof for the unweighted case, as given in §2.7.2. □

2.9.8 Choosing the Objective Function

The objective function used for NNMA plays many different roles. If one has an *a priori* assumption on the distribution of the noise corrupting the observed data, then minimizing the Bregman divergence corresponding to the assumed noise distribution is expected to give a better reconstruction. In fact Monga [200] reported improved empirical results with a particular choice of a Bregman divergence while applying NNMA to an application in image and audio processing. Assuming the noise to be of a Gaussian nature[3], we may prefer to minimize the Frobenius norm, while for Poisson noise, it is more preferable to minimize the KL-Divergence as was also originally done by Lee and Seung [174]. Banerjee et al. [15] illustrate clustering results on data following different distributions, demonstrating that if one the matching Bregman divergence the resulting clustering accuracy is higher, which provides further support to the suggestion of selecting an appropriate divergence based on the noise model.

The particular application at hand can also govern the selection of the objective function, and an example of this can be found in the recent work of Chen et al. [47], Cichocki et al. [50, 52]. From a broader perspective, usually domain knowledge or a better idea about the nature of the input data plays a strong role in governing the choice of divergence.

A final concern, which often plays a significant role in selecting a par-

[3]This connection is approximate because the basic NNMA model admits only non-negative noise.

ticular divergence, especially for large scale data, is the ease with which a particular objective function can be minimized. The Frobenius norm based updates seem to yield the simplest algorithms for NNMA, whereas the other divergences lead to somewhat more complicated updates. The sparsity pattern can also play a role here, for e.g., it might not be possible to fully exploit the sparsity of the data with all the divergences. Furthermore, different divergences have differing characteristics for the solution that they produce. In general, just as selecting an appropriate kernel for kernel based learning algorithms is not always easy, it is difficult to give general prescriptions for which particular divergence measure will be most suited to a given problem.

2.10 Experiments

Now we turn to some experiments with our algorithms to explore how they work in practice. In the first set of our experiments below (§2.10.1) we do not focus on any particular application and point the interested reader to the vast list of applications in Section 2.11.2. In our second set of experiments (§2.10.2) we compare different NNMA objective functions for the task of discovering latent topics within a collection of text documents. In our last set of experiments (§2.10.3) we show how the sparsity of the approximations behaves with differing objective function values.

2.10.1 Monotonic convergence

First we illustrate the monotonic convergence behavior of some of our

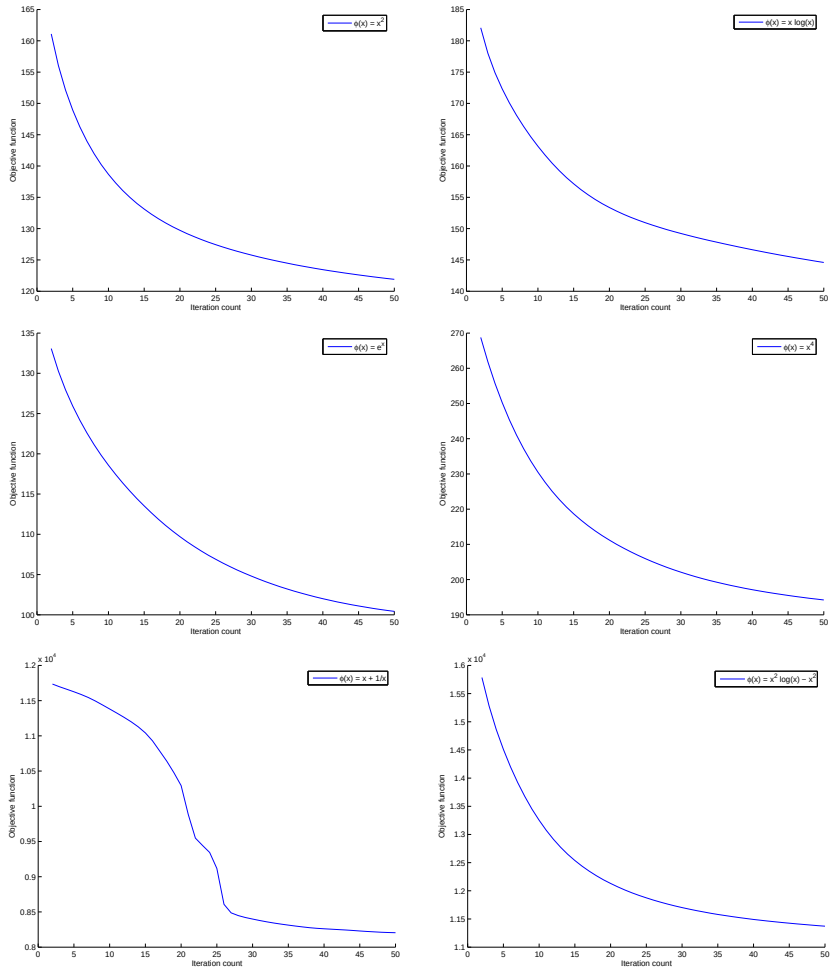


Figure 2.1: Monotonic descent in objective function value for different NNMA problems. From left to right and top to bottom the figures correspond to the results for our Bregman divergence NNMA algorithms corresponding to $\varphi(x) = x^2$, $x \log x$, e^x, x^4, $x + \frac{1}{x}$, $x(x \log x - x)$, respectively. The actual values of the objective functions are less important than their monotonic nature.

algorithms that were implemented in MATLAB. However, this implementation is only for illustrative purposes; we refer the reader to our high performance implementation in C++ for tackling real world datasets.

Least-squares			KL-Divergence		
boundary	pressure	library	patients	boundary	library
layer	mach	system	cells	layer	system
heat	wing	libraries	cases	pressure	retrieval
laminar	shock	retrieval	growth	heat	scientific
transfer	supersonic	systems	lift	shock	systems
plate	jet	research	treatment	mach	science
turbulent	body	science	normal	transfer	research
temperature	theory	scientific	blood	theory	computer
wall	lift	book	cancer	supersonic	language
velocity	bodies	computer	cell	method	buckling

Table 2.2: Top 10 words for the Classic3 dataset using least-squares NNMA and KL-Divergence NNMA

We report monotonicity results for the Bregman divergences generated by the following set of functions (choices of $\varphi(x)$)

$$\varphi(x) = x^2, \quad \varphi(x) = x \log x, \qquad \varphi(x) = e^x,$$
$$\varphi(x) = x^4, \quad \varphi(x) = x + \frac{1}{x}, \quad \varphi(x) = x(x \log x - x).$$

Figure 2.1 shows how the objective function values decrease monotonically for these set of divergences. The algorithm used was an instantiation of our general Bregman divergence NNMA procedure (2.32).

2.10.2 Application to Topic Modeling

Here we show a brief application of NNMA to topic modeling. We use the well-known Classic3 dataset, which is a collection of documents drawn from three underlying categories called Cran, Cisi, and Med. These categories cover aeronautical abstracts, documents from information retrieval and medline, respectively (also see §6.6.1). We ran a rank-3 NNMA approximation on this data set, and we report below the topics discovered by different NNMA algorithms.

From Table 2.2 it can be seen that the least-squares NNMA yields much

Bregman: $\varphi(x) = e^x$			Bregman: $\varphi(x) = x^4$		
pressure	library	boundary	layer	boundary	layers
wing	system	layer	heat	layer	boundary
mach	libraries	heat	laminar	incompressible	laminar
jet	retrieval	laminar	transfer	solutions	turbulent
supersonic	systems	transfer	boundary	shear	wall
lift	research	plate	compressible	equations	layer
body	science	turbulent	shock	velocity	behaviour
shock	scientific	temperature	viscous	plate	separated
wings	book	wall	gradient	turbulent	separation
theory	computer	velocity	hypersonic	approximate	conditions

Table 2.3: Top 10 words for the Classic3 dataset using NNMA with $\varphi(x) = e^x$, and $\varphi(x) = x^4$.

Bregman: $\varphi(x) = x + \frac{1}{x}$			Bregman:$\varphi(x) = x(x \log x - x)$		
differs	services	fit	buckling	library	scientific
breast	formal	language	shells	libraries	wing
deviation	schemes	populations	theory	system	science
naval	makes	individuals	pressure	systems	growth
psychiatric	cleverdon	angles	cylinders	book	lift
notes	courses	proportional	cylindrical	retrieval	wings
considerations	governmental	deviations	flutter	university	literature
women	recording	line	thin	services	journals
strains	turning	providing	indexing	computer	research
processed	analogy	transformation	stress	research	scientists

Table 2.4: Top 10 words for the Classic3 dataset using NNMA with $\varphi(x) = x + \frac{1}{x}$, and $\varphi(x) = x(x \log x - x)$.

less well separated topics than the KL-Divergence version, which seems to have identified the underlying categories. This example illustrates the importance of choosing an objective function while using NNMA in an actual application.

Table 2.3 compares the results of running our Bregman NNMA algorithms (derived as special cases of update (2.32)) using divergences that grow more rapidly than the least-squares case (we used $\varphi(x) = x^4$ and $\varphi(x) = e^x$. From Table 2.3 one observes that using the exponential based loss, only one topic seems to be dominant and occurs almost all three times, whereas for the quartic loss (with $\varphi(x) = x^4$) the results are very similar to those for the least-squares NNMA.

Table 2.4 shows the topics recovered by using two very non-standard

divergence measures. The first divergence measure was generated by using $\varphi(x) = x + \frac{1}{x}$, and it leads to topics with keywords very different from those yielded by other divergences. In fact, it seems that this divergence leads to honing in onto subtopics. The topics seem meaningful because the words within one topic seem well correlated with each other (by observation). The second divergence also has a very interesting behavior. For example, in the second last column of Table 2.4 we see that words and their plurals have been put together into one topic. None of the other divergences shown so far have exhibited this characteristic so clearly. We also remark that since $\varphi''(x) = 2\log x + 1$ for the last objective function considered, we had to scale the data differently to ensure $\varphi''(x) > 0$.

2.10.3 Sparsity of results

For the six different divergence measures we show below the table of sparsity of the factor matrices $\boldsymbol{B}$ and $\boldsymbol{C}$. These sparsity values are somewhat smaller than what we expect, because the NNMA algorithms were run for only 20 iterations each. The aim is to see how rapidly one achieves sparsity within the factors. Table 2.5 highlights how strongly the divergence affects the sparsity of the factor matrices $\boldsymbol{B}$ and $\boldsymbol{C}$. This experiment provides a guideline for the selection of an objective function if good reconstruction and high sparsity are the underlying goals. However, as we see for the topic detection application in the previous section, good sparsity does not necessarily translate into better qualitative results. We remark that for the values reported for NNMA

$\varphi(x)$	nzeros($\boldsymbol{B}$)/(MK)	nzeros($\boldsymbol{C}$)/(KN)	$\|\boldsymbol{A}-\boldsymbol{BC}\|_{\mathrm{F}}/\|A\|_{\mathrm{F}}$
x^2	0.164459	0.050458	0.981300
$x\log x$	0.186614	0.036837	0.985240
e^x	0.185994	0.029812	0.981313
x^4	0.925246	0.893858	0.994716
$x+\frac{1}{x+\epsilon}$	0.969401	0.932836	1.000620
$x(x\log x-x)$	0.192656	0.019361	0.981664

Table 2.5: Sparsity achieved by various NNMA algorithms on the Classic3 dataset. The data size was $(M,N)=(4303,3891)$, and rank $K=3$ approximation was computed. In the table above, nzeros denotes the number of elements smaller than 10^{-6} in the specified matrix. To permit a comparison of the reconstruction between the various algorithms, we show the relative error of reconstruction as measured by the Frobenius norm in the last column. This value must be viewed merely as a guideline, because each algorithm minimizes a different divergence, and only the first one minimizes the Frobenius distance.

corresponding to $\varphi(x)=x+\frac{1}{x+\epsilon}$, we selected $\epsilon=10^{-6}$ to prevent divide by zero errors due to the highly sparse input matrix $\boldsymbol{A}$. From the table we see that both the $\varphi(x)=x^4$ and the $\varphi(x)=x+\frac{1}{x+\epsilon}$ algorithms achieve very high sparsity. However, the former also achieves high reconstruction, as well as very good sparsity.

2.11 Brief Literature Review

Since its introduction, NNMA has been increasingly applied as a technique for dimensionality reduction and data analysis. Correspondingly, there has been a significant amount of research related to it. The aim of this section is to provide a brief summary about the various algorithms and applications of NNMA that have appeared in the literature. While attempt has been made to be as complete as possible, the sheer magnitude of the task renders it impos-

sible to attain completeness. We apologize in advance to the authors whose work we might have inadvertently missed.

The origin of the *approximate* nonnegative factorization problem or NNMA may be credited to [215] who called it Positive Matrix Factorization (PMF), and to [174] who called it Nonnegative Matrix Factorization. The *exact* factorization problem is however older, and Section 2.12 discusses it briefly.

2.11.1 Algorithms

There exist a few different algorithms for NNMA. Some of them are based on solving suitably modified non-linear least squares problems, while others are simple iterative procedures. We summarize procedures of both types below.

2.11.1.1 Paatero's methods

Paatero et al. [215] introduced the term PMF and sought to construct a factor model with two nonnegative matrices by minimizing

$$\|\boldsymbol{W} \odot (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C})\|_{\mathrm{F}}^2, \tag{2.58}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{W}$ are all nonnegative. The matrix $\boldsymbol{W}$ consists of weights reflecting confidence in the measurements in $\boldsymbol{A}$. In the same paper Paatero et al. [215] also introduced a three factor NNMA model. However, they did not provide any algorithm to actually compute the presented models. Paatero and Tapper [213] suggested using alternating least squares (ALS),

wherein one holds $\boldsymbol{B}$ fixed while obtaining the optimal $\boldsymbol{C}$ and vice versa, for PMF. Nonnegativity is enforced in an ad-hoc fashion by simply discarding the entries smaller than zero. NNMA may also be performed with alternating non-negative least squares instead of ALS by using the NNLS algorithm of [170]. While doing ALS or Alternating NNLS, the least squares subroutines can prove to be a bottleneck. Hence, in practice it is better to combine the least square approach with the faster Lee/Seung type updates.

Later [214] proposed another approach for PMF, claiming it to be superior to the one based on ALS. In this approach one iteratively solves $(\boldsymbol{B} + \Delta\boldsymbol{B})\boldsymbol{C} \approx \boldsymbol{A}$ for $\Delta\boldsymbol{B}$ (likewise for $\Delta\boldsymbol{C}$), followed by solving for the coefficient α in $(\boldsymbol{B} + \Delta\boldsymbol{B})(\boldsymbol{C} + \Delta\boldsymbol{C}) \approx \boldsymbol{A}$. However, in practice Paatero and Tapper [214] recommend neglecting the product $\Delta\boldsymbol{B}\Delta\boldsymbol{C}$ while minimizing $\|\boldsymbol{A} - (\boldsymbol{B} + \Delta\boldsymbol{B})(\boldsymbol{C} + \Delta\boldsymbol{C})\|_{\mathrm{F}}$ to obtain $\Delta\boldsymbol{B}$ and $\Delta\boldsymbol{C}$. In [210], yet another algorithm for PMF is introduced under the name PMF2 (the two standing for a two-factor model). However, from its description, the PMF2 algorithm seems to have expanded upon the just described method of [214] and it enforces nonnegativity using logarithmic penalty functions.

Paatero [211] went on to consider a three-way factor analytic model (also called PARAFAC, a factor model introduced in 1970 by Harshman [120]). The corresponding algorithm for computing nonnegative factors was called PMF3 and pseudocode is provided in the paper [211]. However, the algorithm requires a significant amount of engineering effort to implement and is rather obscure. As an application to the same problem (PARAFAC) Bro and

de Jong [40] presented a faster NNLS algorithm. In order to solve more general "multi-factor" problems Paatero [212] developed another algorithm called the Multi-linear Engine (ME) that allows solving n-way models. The solution is computed using a method based on conjugate gradients.

Other methods based on Least Squares Pauca et al. [223] presented an algorithm that combines a constrained least squares problem with the multiplicative update procedures of Lee and Seung [174]. The procedure solves the least square problem $\min \|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2 + \lambda\|\boldsymbol{C}\|_{\mathrm{F}}^2$ using ordinary least squares. The nonnegativity of $\boldsymbol{C}$ is enforced by setting the negative elements to 0. The matrix $\boldsymbol{B}$ is updated using the standard updates (§ 2.9.5). Langville and Meyer [165] suggest using alternating constrained least squares for both $\boldsymbol{B}$ and $\boldsymbol{C}$. The λ term influences the sparsity of the resulting solution. Langville and Meyer [165] also discuss other measures of sparsity that one could incorporate. Other related work dealing with sparsity in NNMA is [216] (controlling rotations by influencing sparsity) and [125] (for nonnegative tensors). In a vein similar to alternating NNLS Lawrence et al. [168] describe an alternating constrained nonnegative least squares procedure for NNMA built on top of linearly constrained least squares. Lin [182] describes two projected gradient approaches, one directly applied to the matrix formulation of NNMA and the other to using the projected gradient method for solving the individual NNLS sub-problems. The brief survey paper [23] describes some other approaches for solving the Frobenius norm NNMA problem.

More recently Kim et al. [153] have developed a projected Quasi-Newton method for solving the least-squares NNMA problem. Their method derives an efficient method for solving the NNLS problem and alternatingly applies it to optimize over $\boldsymbol{B}$ and $\boldsymbol{C}$. For most problems, this method yields results superior to the traditional Lee & Seung NNMA method.

2.11.1.2 Lee & Seung and Related Methods

Lee and Seung also developed the problem of NNMA and introduced a specially constrained version of it in the context of unsupervised learning by convex and conic coding [173]. In that paper, they considered learning encodings so that the reconstruction error over the ensemble of inputs is minimized. The method of choice was an alternating projected gradient approach in which first $\boldsymbol{B}$ is fixed and a gradient descent is done w.r.t. $\boldsymbol{C}$ and vice versa. Nonnegativity constraints were implemented by zeroing out the negative entries and the normalization constraints were enforced using quadratic penalty functions. However, NNMA finally gained popularity after the two papers [172, 174] introduced the problem under the name *nonnegative matrix factorization*. Lee and Seung [174] provided efficient iterative algorithms for NNMA, which were developed and analyzed further in [172].

Hoyer [131] added an ℓ_1-norm based regularization term to the original Frobenius norm objective function in order achieve sparser solutions. The

resultant NNMA problem, which he named Nonnegative Sparse Coding, was

$$\min_{\boldsymbol{B},\boldsymbol{C}\geq 0} \|\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2 + \lambda \sum_{ij} c_{ij},$$

where $\lambda > 0$ is a regularization parameter. Subsequently, Hoyer [133] extended the enforcement of sparsity by minimizing $\|\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2$ under additional sparsity constraints of the form $\mathrm{sparsity}(\boldsymbol{c}_j^T) = S_C$, $\mathrm{sparsity}(\boldsymbol{b}_i) = S_B$. Hoyer [133] uses the function

$$\mathrm{sparsity}(\boldsymbol{x}) = \frac{\sqrt{n} - \|\boldsymbol{x}\|_1/\|\boldsymbol{x}\|_2}{\sqrt{n}-1},$$

to measure the sparsity and uses a combination of projected gradient descent and Lee/Seung's iterative updates for carrying out the minimization. Evidently, one can use other measures of sparsity (See [165], for further examples).

Feng et al. [95] added additional constraints to the KL-Divergence NNMA problem to model spatial locality in the input matrix $\boldsymbol{A}$. Locality is encouraged by enforcing constraints on $\boldsymbol{B}$, and sparsity by imposing constraints on $\boldsymbol{C}$. The resultant objective function was

$$\mathrm{KL}(\boldsymbol{A};\boldsymbol{B}\boldsymbol{C}) + c_1\mathbf{1}^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{1} - c_2\|\boldsymbol{C}\|_{\mathrm{F}}^2, \tag{2.59}$$

where $c_1, c_2 > 0$ are some constants.

Sajda et al. [245] modified Lee/Seung's algorithm by forcing small values in $\boldsymbol{C}$ to $\epsilon > 0$, and named their modification cNMF (constrained NMF). They initialized $\boldsymbol{B}$ randomly, and $\boldsymbol{C}$ using a constrained least squares solution. Thereafter, they updated $\boldsymbol{B}$ and $\boldsymbol{C}$ as usual with the exception of clamping down small values in $\boldsymbol{C}$ to the fixed constant ϵ.

Guillamet et al. [115, 116] suggest that one should weight the input vectors (columns of $\boldsymbol{A}$) and consider the approximation $\boldsymbol{AW} \approx \boldsymbol{BCW}$, where $\boldsymbol{W}$ is a diagonal matrix of weights such that $\mathrm{Tr}(\boldsymbol{W}) = 1$. They present results for such a modification to the KL-Divergence NNMA problem. Our weighted NNMA described in Section 2.9.7 subsumes this approach.

Szatmáry et al. [270] perform NNMA that has been augmented with sparse code shrinkage and weight sparsification. The latter two techniques were employed to improve the performance of NNMA. For more on SCS the reader is referred to [137]. Heiler and Schnörr [126] use NNMA and second-order cone programming to obtain sparse representations.

The NNMA problem has been extended to nonnegative approximations for tensors. Welling and Weber [286] derive algorithms similar to the iterative Lee/Seung schemes for minimizing squared and KL-Divergence losses (for tensors). Shashua and Hazan [258] perform nonnegative tensor factorization by repeated rank-1 approximations, while minimizing a squared loss objective function. They include a proof of convergence of their procedure. Heiler and Schnörr [125] study sparseness in the context of NTF.

New methods for minimizing Csiszar's divergence are described by Cichocki et al. [51, 52]. NNMA using quasi-Newton methods is considered by Zdunek and Cichocki [296], who apply it to Amari's α-disparity [2]. Cichocki et al. [50] also derive other iterative methods for minimizing Amari's α-divergence.

2.11.2 Applications

We now enlist some of the numerous applications of NNMA that have appeared in the literature. We have roughly categorized them for easier perusal. Some of the applications are divergent from a traditional machine learning setting, but as the original PMF series of algorithms arose in such applications, we have decided to retain references to them for completeness.

2.11.2.1 Environmetrics and Chemometrics

Paatero et al. applied the ideas of PMF to environmental data as early as 1991. For a list of references that indicate some of these applications the reader is referred to the original PMF paper [214]. Later Paatero [212] applied his multi-linear engine to analyze atmospheric emission and pollution data. A paper discussing the application of orthogonal projection approach, alternating least squares and PMF to analyze chromatographic spectral data (which is used to analyze mixtures of chemicals) was presented by Frenich et al. [99]. The results obtained by these three methods are compared by evaluating measures of dissimilarity between real and estimated spectra (matrix $\boldsymbol{C}$). The authors concluded that in general PMF2 and alternating least squares had little differences in the quality of results, and that PMF2 is a good tool for curve resolution analysis of chromatographic data. Qin et al. [231] used PMF on a large aerosol database measured in Hong Kong incorporating error estimates through the $\boldsymbol{W}$ matrix.

Paatero et al. [216] discuss the resolution of the problem of rotational indeterminacy in the PMF (PMF2, PMF3, ME) solutions using a specific two factor model as an example. The conclusions and recommendations of the paper are however, largely empirical in nature. Ramadan et al. [232] compare PMF and the ME on a data matrix of pollutant concentrations in Phoenix, and they conclude that the ME did not yield significant modeling advantages over PMF2. Sajda et al. [245] applied their constrained version of NNMA to recovering constituent spectra in 3D chemical shift imaging. They compared their results to Bayesian Spectral Decomposition [208] and suggested that NNMA obtains similar results in orders of magnitude lesser time.

2.11.2.2 Image Processing and Computer Graphics

In their seminal paper Lee and Seung [174] demonstrated how one could obtain a parts based representation for image data. That is, the sparse basis vectors (columns of $\boldsymbol{B}$) approximating faces roughly corresponded to individual parts of faces such as lips, noses and eyes. Feng et al. [95] used their local NNMA algorithm for learning a spatially localized, parts-based representation for images. They compare their method to PCA and NNMA to demonstrate the situations where a spatially localized approach has advantages (such as highly occluded faces during face recognition). Guillamet and Vitrià [112] suggest using the Earth Movers Distance as a relevant metric for doing face recognition using NNMA. Other work on face and image processing applications of NNMA by these authors includes [113–116]. Cooper and Foote [56]

applied NNMA to summarizing video and audio data.

Wild et al. [287] described an application of NNMA to Airborne Visible/Infrared Imaging Spectrometer data. They describe feature extraction using a random initialization of NNMA as well as via an initialization based on a spherical kmeans clustering. Szatmáry et al. [271] proposed hierarchical image representation using NNMA augmented with sparse code shrinkage preprocessing and applied their methods to the FERET image database. Other image processing work that uses NNMA includes [163, 168, 169, 297]. The recent article of Spratling [263] evaluates the empirical performance of some NNMA algorithms for recognizing elementary image features, especially in the presence of occlusion.

Nonnegative tensor factorization (NTF) was used by Welling and Weber [286] to the decomposition of color images. Shashua and Hazan [124, 258] applied NTF to low-rank representation of images, obtaining good parts based representations.

2.11.2.3 Text analysis

Lee and Seung [174] applied NNMA to text documents and highlighted the ability of NNMA to tackle semantic issues such as synonymy. Owing to the low-rank approximations produced NNMA is a natural candidate for a clustering procedure. Xu et al. [293] described clustering experiments with NNMA, wherein they compared NNMA against spectral methods, suggesting that the former can obtain higher accuracy. Xu et al. [293] used NNMA

for clustering text data. Other related work on clustering and text analysis using NNMA includes [5, 223, 255]. An application to email surveillance was discussed in [25],

2.11.2.4 Blind Source Separation & ICA

Some authors have considered blind source separation by using either nonnegative PCA [209] or ICA [229, 230]. Work that directly applies NNMA to blind source separation and ICA includes Cichocki et al. [52], Li and Cichocki [179]. Pauca et al. [222] use NNMA and ICA for unmixing data.

2.11.2.5 Bioinformatics

Recently various data mining techniques have been applied to problems or data sets from biology forming a significant part of the field of bioinformatics. NNMA has had its share of applications. Brunet et al. [41] apply NNMA to form *metagenes* to infer biological information from cancer-related microarray data. They use the KL-Divergence based NNMA algorithm and also provide heuristic methods for model selection. Kim and Tidor [154] apply NNMA for performing dimensionality reduction to aid in the identification of subsystems from gene microarray data. They hinged their arguments on the ability to detect local features from the data using NNMA. Other applications include lung cancer prognosis [141], analysis of lung cancer profiles [100], sparse NNMA for cancer class discovery [101], among others. Further references that apply NNMA or sparse variants thereof, to gene data are [6, 221, 234]. Chen

et al. [47] apply their NNMA algorithms to the analysis of data related to Alzheimer's disease.

2.11.2.6 Miscellaneous applications

NNMA has been applied to problems of a diverse nature. Though we summarized some of the major applications above, there remain numerous other applications. We cannot hope to be exhaustive in our coverage and must thereby satisfy ourselves by being indicative. Hoyer [131] added sparsity constraints to NNMA and in a later paper [132] modeled the receptive fields of the primary visual cortex in mammals. Hoyer's experiments on natural images revealed the usefulness of an NNMA based approach.

Behnke [19] proposed a variant of NNMA called convolutional NNMA and applied it to a hierarchical approach for extracting speech features. NNMA was combined with a Neural Abstraction Pyramid architecture [18] and recursively applied to to obtain a hierarchical decomposition of the features.

A somewhat offbeat application to the transcription of polyphonic music via NNMA was attempted by Smaragdis and Brown [262], who analyzed polyphonic music passages that comprised of notes that exhibit a harmonically fixed spectral profile.

J-H. Ahn and Choi [144], Lee et al. [176] apply NNMA to the analysis of matrices obtained via dynamic Positron Emission Tomography (PET). The ability to use a Poisson statistics based noise model for NNMA for PET images is suggested to be one of the benefits of NNMA over traditional Gaussian based

methods since PET data comes from a process where the Poisson distribution makes more sense. This motivation also lies behind using an appropriate Bregman divergence for an NNMA problem depending on the assumed underlying nature of the noise distribution.

Other applications include object characterization [227], spectral data analysis [224], learning sound dictionaries [4], mining ratio-rules [134], and multiway clustering [5, 259].

2.12 Nonnegative Matrix Factorization

For completeness (and to ratify our selection of the name NNMA) we digress briefly to describe the nonnegative matrix factorization problem, i.e., an NNMA problem where an exact factorization of the form $\boldsymbol{A} = \boldsymbol{BC}$ exists. We provide only a smattering of references to this problem, hopefully pointing the interested reader in the correct direction.

Markham [189] derived necessary and sufficient conditions for a nonnegative matrix $\boldsymbol{A}$ to have a factorization of the form $\boldsymbol{LU}$, where $\boldsymbol{L}$ is nonnegative lower triangular and $\boldsymbol{U}$ is a nonnegative unit upper triangular matrix. He restricted $\boldsymbol{A}$ to the class of matrices that have nonzero principal subminors. This somewhat artificial restriction was lifted in a subsequent paper [166]. Related work discussing "correct" decomposition into parts may be found in a more recent paper [90]. Markham has also discussed factorizations of completely positive matrices, i.e., matrices all of whose minors are positive [188]. Later Cryer [60] proved that a matrix $\boldsymbol{A}$ is strictly totally positive iff $\boldsymbol{A} = \boldsymbol{LU}$, where

$\boldsymbol{L}$ and $\boldsymbol{U}$ are triangular matrices all of whose non-trivial minors are strictly positive. Other relevant references include [118, 150, 180].

Gray and Wilson [110] provided geometric proofs of the fact that for $n \leq 4$, $n \times n$ nonnegative positive-definite matrices can be factored into $n \times n$ nonnegative factors. They also show that their conditions are not sufficient to guarantee the existence of such factorizations for $n \geq 5$.

Suppose $\boldsymbol{A}$ is an $m \times n$ matrix of rank $r \leq \min(m, n)$. Then, $\boldsymbol{BC}$ is called a *rank factorization* of $\boldsymbol{A}$ if $\boldsymbol{B}$ and $\boldsymbol{C}$ are $m \times r$, $n \times r$ full-rank matrices, and $\boldsymbol{A} = \boldsymbol{BC}$. Of course, for a nonnegative rank factorization (NRF) both $\boldsymbol{B}$ and $\boldsymbol{C}$ are nonnegative. Campbell and Poole [43] discuss the existence of generalized matrix inverses in terms of NRFs. They also present an algorithm that can compute a NRF of a nonnegative matrix when a nonnegative 1-inverse exists[4]. Thomas [273] gave a simple characterization when a NRF exists for a given matrix. Wall [280] discusses rank factorizations of positive operators. Jeter and Pye [147] prove that if $\boldsymbol{A}$ is weakly monotone [22] then it has a NRF if and only if it possesses an $r \times r$ monomial submatrix. Chen [46] describes when $\boldsymbol{A}$ has "trivial" or "non-trivial" NRFs.

[4] A matrix $\boldsymbol{X}$ is called a 1-inverse of $\boldsymbol{A}$ if $\boldsymbol{AXA} = \boldsymbol{A}$ and a 2-inverse if $\boldsymbol{XAX} = \boldsymbol{X}$.

Chapter 3

Incremental Low Rank Matrix Approximation

In the previous chapter we studied non-negative matrix approximations of the form $\boldsymbol{A} \approx \boldsymbol{BC}$, where starting from some initial values, the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ were iteratively updated to compute the said approximation. In this chapter, we build upon the algorithmic techniques of the previous chapter to develop methods that estimate $\boldsymbol{B}$ and $\boldsymbol{C}$ incrementally, i.e., one row or column at a time.

Incremental approximations are particularly attractive in the face of large volumes of data because they usually lead to simple and scalable algorithms. Furthermore, model selection (e.g., rank) is much more flexible in an incremental setting. The incremental approach is also more robust than a corresponding batch method since it has fewer degrees of freedom, i.e., the number of parameters it must update at each iteration is much smaller than for a batch approach, which limits overfitting at each step. Finally, incremental matrix approximation takes an important step towards truly online low-rank approximations. Incremental matrix approximations are of course, not that surprising. The well-known Singular Value Decomposition (SVD) is in practice computed using an (block) incremental Lanczos procedure.

3.1 Problem Formulation

As before, we assume the input data is available as a set of vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$, where each $\boldsymbol{a}_i \in \mathbb{R}^M$. We collect these N vectors into an $M \times N$ matrix $\boldsymbol{A}$ for which a reduced dimension representation may be written as

$$\boldsymbol{A} \approx \hat{\boldsymbol{A}} = \sum_{k=1}^{K} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T, \tag{3.1}$$

where $\boldsymbol{u}_k \in \mathbb{R}^M$, $\boldsymbol{v}_k \in \mathbb{R}^N$, and $\sigma_k \in \mathbb{R}$. Usually, $K \ll (M, N)$ for low-rank approximation problems. Our notation has been selected to provide an analogy with the well known SVD based approximation. Naturally, the quantities $\boldsymbol{u}_k$, $\boldsymbol{v}_k$, and σ_k will behave differently and have various interesting characteristics contingent upon the objective function used to measure the quality of approximation in (3.1).

Let Δ be some differentiable proper convex function. Initially, for ease of exposition, we restrict our attention to convex functions $\Delta : S \subseteq \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ that are at least convex in one of their arguments. Other restrictions or changes to Δ will be mentioned as needed. For simplicity, we restrict our attention to separable distortion functions. Thus, we measure the error of approximation in (3.1) by

$$\Delta(\hat{\boldsymbol{A}}, \boldsymbol{A}) = \sum_{ij} \nu_{ij} \Delta(\hat{a}_{ij}, a_{ij}), \tag{3.2}$$

where the $\nu_{ij} \geq 0$ are pre-specified weights. In an incremental setting, we assume that a rank-$(K-1)$ approximation to $\boldsymbol{A}$ is already available, and we augment it to obtain a rank-K approximation. In symbols, if $\boldsymbol{T} = \sum_{k=1}^{K-1} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T$,

gives the rank-$K-1$ approximation, then $\hat{\boldsymbol{A}} = \boldsymbol{T} + \sigma_K \boldsymbol{u}_K \boldsymbol{v}_K^T$ gives a rank-K approximation.[1] Given $\boldsymbol{T}$, we can compute σ_K, $\boldsymbol{u}_K$, and $\boldsymbol{v}_K$ by solving the following regularized optimization problem (we drop the subscript K for brevity),

$$\min_{\sigma, \boldsymbol{u}, \boldsymbol{v}} \ \Delta(\boldsymbol{T} + \sigma \boldsymbol{u}\boldsymbol{v}^T, \boldsymbol{A}) + \lambda R_u(\boldsymbol{u}) + \omega R_v(\boldsymbol{v}), \tag{3.3}$$

where $\lambda, \omega \geq 0$ determine the influence of the regularization functions R_u and R_v, which themselves are assumed to be strictly convex and differentiable (see Section 3.5 for a discussion of the case where R_u and R_v are concave). Problem (3.3) encompasses as special cases incremental SVD, incremental PLSI, and incremental regularized aspect models, in addition to providing incremental formulations for a host of other dimensionality reduction problems—see Section 3.3 for details. We stress at this point that even though Δ is convex, it is not necessarily simultaneously convex in $\sigma \boldsymbol{u}\boldsymbol{v}^T$, hence one cannot hope to obtain algorithms that achieve global optima of (3.2), and we have to content ourselves with at most locally optimal solutions. Note that if $\sigma \boldsymbol{u}\boldsymbol{v}^T$ is a solution, then so is $\sigma(c\boldsymbol{u})(\frac{1}{c}\boldsymbol{v})$ for any non-zero constant c. Hence, we normalize $\boldsymbol{u}$ and $\boldsymbol{v}$, and adjust σ accordingly. In Section 3.2 we derive generic algorithms for solving (3.3).

[1] In some circumstances it is possible that $\boldsymbol{T} + \sigma_K \boldsymbol{u}_K \boldsymbol{v}_K^T$ has rank lesser than $\boldsymbol{T}$. However, for simplicity we do not consider this situation.

3.1.1 A convex variation of (3.1)

When building low-rank models that need to have probabilistic interpretations, it makes more sense to construct approximations of the form

$$\hat{\boldsymbol{A}} = (1-\alpha)\boldsymbol{T} + \alpha(\sigma \boldsymbol{u}\boldsymbol{v}^T), \tag{3.4}$$

where $\alpha \in [0,1]$. The optimization problem (3.3) is adjusted accordingly. Such a convex combination of $\boldsymbol{T}$ and $\sigma\boldsymbol{u}\boldsymbol{v}^T$ may be interpreted as approximating an underlying joint distribution, where α and $(1-\alpha)$ act as a priors for $\sigma\boldsymbol{u}\boldsymbol{v}^T$ and $\boldsymbol{T}$, respectively. This viewpoint was originally motivated by the vertex direction method from semi-parametric mixture estimation [31], and was also exploited in a supervised setting by AnyBoost [190]. Our formulation also deals with these cases with equal ease and additionally permits the imposition of regularization, which can be of practical importance. For example, in the incremental density boosting scenario, a common problem is the repeated rediscovery of an already discovered model (see [269] for related discussion). To prevent this, one can add entropic or Euclidean regularization terms that force the new model $\sigma\boldsymbol{u}\boldsymbol{v}^T$ to differ from the previous model $\boldsymbol{T}$.

3.2 Algorithms

A simple approach for solving (3.3) involves computing the derivatives of Δ w.r.t. σ, $\boldsymbol{u}$, and $\boldsymbol{v}$, setting them to zero, and solving the resultant equations (see Example 3.4). However, often these equations are not easy to solve. Potentially one can solve for u_i, v_j and σ using one-dimensional minimization,

but for large problems such an approach can be extremely prohibitive. We seek simpler and consequently more efficient iterative algorithms, which guarantee descent on the objective function.

3.2.1 Generic methods for (3.3)

We begin with generic methods for Problem (3.3). Two basic ingredients form the core of our derivations, namely, the method of auxiliary functions and the convexity of Δ. Auxiliary functions have appeared in various guises with different cognomens such as surrogate or transfer functions, majorization functions, and variational bounds [164, 294]. Additionally, see Chapter 2 for an extensive use of auxiliary functions for solving the NNMA problem. For reference, we restate the definition below.

Definition 3.1 (Auxiliary function)**.** Let $F : S \to \mathbb{R}$, and $G : S \times S \to \mathbb{R}$. The function G is called an *auxiliary function* for F if

1. $F(x) \leq G(x, y)$ for all x, y in S.
2. $F(x) = G(x, x)$ for all x in S.

The auxiliary function G is constructed in a manner that makes it easier to minimize than F. This ease is usually obtained by designing a G that decouples the t_{ij} (components of $\boldsymbol{T}$) from $\sigma u_i v_j$. Note that for computing a rank-one approximation, no such decoupling is needed. We exploit the

convexity of Δ to construct G.[2] Consider the distortion $\Delta(\hat{a}_{ij}, a_{ij})$; we have

$$\begin{aligned}&\Delta(t_{ij} + \sigma u_i v_j, a_{ij}) \\ \leq&(1-\beta_{ij})\Delta\left(\frac{t_{ij}}{(1-\beta_{ij})}, a_{ij}\right) + \beta_{ij}\Delta\left(\frac{\sigma u_i v_j}{\beta_{ij}}, a_{ij}\right),\end{aligned}$$

where we define the convex coefficient

$$\beta_{ij} = \frac{|\tilde{\sigma}\tilde{u}_i\tilde{v}_j|}{|t_{ij}| + |\tilde{\sigma}\tilde{u}_i\tilde{v}_j|}, \tag{3.5}$$

along with auxiliary variables $\tilde{\sigma}$, $\tilde{u}_i$, and $\tilde{v}_j$. Now, using the definition (3.2), we define a function G as

$$\begin{aligned}&G(\sigma, \boldsymbol{u}, \boldsymbol{v}; \tilde{\sigma}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) = \\ &\sum_{ij}\nu_{ij}(1-\beta_{ij})\Delta\left(\frac{t_{ij}}{(1-\beta_{ij})}, a_{ij}\right) + \\ &\nu_{ij}\beta_{ij}\Delta\left(\frac{\sigma u_i v_j}{\beta_{ij}}, a_{ij}\right) + \lambda R_u(\boldsymbol{u}) + \omega R_v(\boldsymbol{v}).\end{aligned} \tag{3.6}$$

Lemma 3.2 shows that G is an auxiliary function.

Lemma 3.2 (Auxiliary function)**.** *Let $F(\sigma, \boldsymbol{u}, \boldsymbol{v})$ denote the objective function in* (3.3)*, i.e.,*

$$F(\sigma, \boldsymbol{u}, \boldsymbol{v}) = \Delta(\boldsymbol{T} + \sigma\boldsymbol{u}\boldsymbol{v}^T) + \lambda R_u(\boldsymbol{u}) + \omega R_v(\boldsymbol{v}).$$

Let $s_{x\geq 0} = +1$, $s_{x<0} = -1$; if $s_{t_{ij}} s_\sigma s_{u_i} s_{v_j} = 1$, then G is an auxiliary function for F.

[2]Exploiting the convexity is one of the possible ways of constructing auxiliary functions. For other interesting methods the reader is referred to [164, 177, 294].

Proof. Clearly due to the convexity of Δ, the inequality $F \leq G$ holds trivially. We only need to verify that $F(\sigma, \boldsymbol{u}, \boldsymbol{v}) = G(\sigma, \boldsymbol{u}, \boldsymbol{v}; \sigma, \boldsymbol{u}, \boldsymbol{v})$. Note that without loss of generality we can ignore the regularization terms, since they are common to both F and G. Therefore, $G(\sigma, \boldsymbol{u}, \boldsymbol{v}; \sigma, \boldsymbol{u}, \boldsymbol{v})$ essentially equals

$$\begin{aligned}
&\sum_{ij} \nu_{ij}(1-\beta_{ij})\Delta\big(s_{t_{ij}}(|t_{ij}| + |\sigma u_i v_j|), a_{ij}\big) + \\
&\qquad \nu_{ij}\beta_{ij}\Delta\big(s_\sigma s_{u_i} s_{v_j}(|t_{ij}| + |\sigma u_i v_j|), a_{ij}\big) \\
&= \sum_{ij} \nu_{ij}(1-\beta_{ij})\Delta\big(t_{ij} + s_{t_{ij}} s_\sigma s_{u_i} s_{v_j} \sigma u_i v_j\big) + \\
&\qquad \nu_{ij}\beta_{ij}\Delta\big(s_{t_{ij}} s_\sigma s_{u_i} s_{v_j} t_{ij} + \sigma u_i v_j, a_{ij}\big) \\
&= \sum_{ij} \nu_{ij}\beta_{ij}\Delta\big(t_{ij} + \sigma u_i v_j\big) \\
&= \quad F(\sigma, \boldsymbol{u}, \boldsymbol{v}),
\end{aligned}$$

where we used the assumption $s_{t_{ij}} s_\sigma s_{u_i} s_{v_j} = 1$ to simplify the last step. □

Remarks: The assumption that $s_{t_{ij}} s_\sigma s_{u_i} s_{v_j} = 1$ holds trivially for problems that use only non-negative data. In general ensuring the validity of this assumption is not always possible. However, for such difficult cases one can still obtain a heuristic procedure for optimizing F, since the inequality $F \leq G$ always holds.

Lemma 3.3 below shows how to use the auxiliary function G to obtain an algorithm that guarantees monotonic descent in F.

Lemma 3.3 (Descent)**.** *Given G as an auxiliary function for F, solving Equations* (3.7a)–(3.7c) *leads to a decrease in F.*

Proof. Since $F \leq G$, we have

$$\begin{aligned}
& F(\sigma^{t+1}, \boldsymbol{u}^{t+1}, \boldsymbol{v}^{t+1}) \\
& = G(\sigma^{t+1}, \boldsymbol{u}^{t+1}, \boldsymbol{v}^{t+1}; \sigma^{t+1}, \boldsymbol{u}^{t+1}, \boldsymbol{v}^{t+1}) \\
& \leq G(\sigma^{t}, \boldsymbol{u}^{t}, \boldsymbol{v}^{t}; \sigma^{t}, \boldsymbol{u}^{t}, \boldsymbol{v}^{t}) \\
& = F(\sigma^{t}, \boldsymbol{u}^{t}, \boldsymbol{v}^{t}).
\end{aligned}$$

The inequality in Line 3 above follows trivially when

$$(\sigma^{t+1}, \boldsymbol{u}^{t+1}, \boldsymbol{v}^{t+1}) = \operatorname*{argmin}_{\sigma, \boldsymbol{u}, \boldsymbol{v}} G(\sigma, \boldsymbol{u}, \boldsymbol{v}; \sigma^{t}, \boldsymbol{u}^{t}, \boldsymbol{v}^{t}),$$

as obtained by solving (3.7a)–(3.7c). □

Minimizing G. Since G decouples t_{ij} and $\sigma u_i v_j$, it is easier to minimize G with respect to the parameters σ, $\boldsymbol{u}$, and $\boldsymbol{v}$. To that end we differentiate G w.r.t. the individual parameters to obtain

$$\frac{\partial G}{\partial \sigma} = \sum_{ij} \nu_{ij} \beta_{ij} \frac{\partial \Delta}{\partial \sigma}\Big(\frac{\sigma u_i v_j}{\beta_{ij}}, a_{ij}\Big) u_i v_j \tag{3.7a}$$

$$\frac{\partial G}{\partial u_p} = \sum_{j} \nu_{pj} \beta_{pj} \frac{\partial \Delta}{\partial u_p}\Big(\frac{\sigma u_p v_j}{\beta_{pj}}, a_{pj}\Big) \sigma v_j + \lambda \frac{\partial R_u(\boldsymbol{u})}{\partial u_p}, \tag{3.7b}$$

$$\frac{\partial G}{\partial v_q} = \sum_{i} \nu_{iq} \beta_{jq} \frac{\partial \Delta}{\partial v_q}\Big(\frac{\sigma u_i v_q}{\beta_{iq}}, a_{iq}\Big) \sigma u_i + \omega \frac{\partial R_v(\boldsymbol{v})}{\partial v_q}. \tag{3.7c}$$

Whether these equations can be solved analytically or not depends upon Δ and R_u, R_v. Nevertheless, in case an analytic solution is not possible, one can solve (3.7a)–(3.7c) iteratively by cycling through the variables σ, $\boldsymbol{u}$, and $\boldsymbol{v}$ (usually a few (2–5) iterations are sufficient).

3.2.2 Generic Methods for (3.4)

In the formulation given by (3.4) we need to estimate the "prior" parameter α in addition to σ, $\boldsymbol{u}$, and $\boldsymbol{v}$. Two approaches can be applied here. First, one can simply do a line-search for α, so that the limited-minimization rule

$$\alpha = \underset{\alpha \in [0,1]}{\operatorname{argmin}} \Delta\big((1-\alpha)\boldsymbol{T} + \alpha(\sigma \boldsymbol{u}\boldsymbol{v}^T)\big), \tag{3.8}$$

is used to obtain the value of α once σ, $\boldsymbol{u}$, and $\boldsymbol{v}$ have been computed using either the auxiliary function technique of Section 3.2.1, or a functional gradient descent approach similar to that used by the AnyBoost framework [190]. In the latter case, one computes a first-order approximation

$$\Delta(\boldsymbol{T} + \alpha(\sigma \boldsymbol{u}\boldsymbol{v}^T - \boldsymbol{T})) \approx \Delta(\boldsymbol{T}) + \alpha \langle \boldsymbol{\nabla}\Delta,\, (\sigma \boldsymbol{u}\boldsymbol{v}^T - \boldsymbol{T}) \rangle,$$

where

$$\boldsymbol{\nabla}\Delta = \frac{\partial \Delta(\hat{\boldsymbol{A}} + \delta \mathbb{1}, \boldsymbol{A})}{\partial \delta}\Big|_{\delta=0},$$

is the functional derivative of Δ and $\mathbb{1}$ is the indicator function for the ij-entry of $\hat{\boldsymbol{A}}$. Now, in order to perform a descent on Δ, we can maximize the directional derivative $\langle \boldsymbol{\nabla}\Delta,\, (\sigma \boldsymbol{u}\boldsymbol{v}^T - \boldsymbol{T}) \rangle$. Notice that for density "boosting" [239], the function $\Delta \equiv -\log$. The above approach is also similar to the Frank-Wolfe algorithm [45] that proceeds by linearizing the initial objective function. The next section provides examples illustrating the use of these principles for deriving algorithms.

3.3 Example Problems

We illustrate a few simple examples in this section to put the generality of our approach in perspective. Each example derives an incremental algorithm for obtaining a low-rank approximation by minimizing a chosen distortion. The resulting method can be applied to tasks that depend on low-rank approximation, for example, clustering, topic modeling, or incremental density estimation, to name a few. Our approach also yields several new incremental algorithms for the NNMA problem studied in Chapter 2.

Example 3.4 shows how an incremental SVD algorithm falls out of our framework, while Example 3.5 imposes regularization on the SVD parameters for obtaining sparser factors. Example 3.6 presents a new problem, which we call *Entropic factorization* because it aims to build low-rank approximations while attempting to approximately maximize the entropy of the estimated parameters. Examples 3.7–3.10 provide various instances of approximations using the KL-Divergence, including regularized versions as well as an algorithm via functional gradients. An entire class of updates is then given by Example 3.11, which derives the updates for minimizing Bregman divergences.

Example 3.4 (Incremental SVD–Naïve method)**.** The SVD is by far the most common matrix decomposition. Here we show how easily our method yields a trivial incremental SVD algorithm. Naturally, this is a naïve method, because usually in practice one uses a block-Lanczos process, which is also an *incremental* method. Let $\Delta(x, y) = \frac{1}{2}(x - y)^2$, $R_u, R_v \equiv 0$, and all $\nu_{ij} = 1$.

Equations (3.7a)–(3.7c) yield the updates

$$\begin{aligned} \boldsymbol{u} &\leftarrow (\boldsymbol{A}-\boldsymbol{T})\boldsymbol{v}; \quad \boldsymbol{u} \leftarrow \frac{\boldsymbol{u}}{\boldsymbol{u}^T\boldsymbol{u}}, \\ \boldsymbol{v} &\leftarrow (\boldsymbol{A}-\boldsymbol{T})^T\boldsymbol{u}; \quad \boldsymbol{v} \leftarrow \frac{\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}}, \end{aligned} \tag{3.9}$$

for $\boldsymbol{u}$, and $\boldsymbol{v}$, while $\sigma = \boldsymbol{u}^T(\boldsymbol{A}-\boldsymbol{T})\boldsymbol{v}$. These are exactly the singular vectors and values of $\boldsymbol{A}-\boldsymbol{T}$.

Example 3.5 (Regularized SVD)**.** We can add the regularization penalties $R_u(\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|^2$, and $R_v(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v}\|^2$ to obtain make the solutions $\boldsymbol{u}$ and $\boldsymbol{v}$ sparser. With $\boldsymbol{M} = \boldsymbol{A}-\boldsymbol{T}$, the iterative updates can be written as

$$\boldsymbol{u} = \frac{\sigma\boldsymbol{M}\boldsymbol{v}}{\sigma^2\boldsymbol{v}^T\boldsymbol{v}+\lambda}; \quad \boldsymbol{v} = \frac{\sigma\boldsymbol{M}^T\boldsymbol{u}}{\sigma^2\boldsymbol{u}^T\boldsymbol{u}+\omega},$$

where $\sigma = \boldsymbol{u}^T\boldsymbol{M}\boldsymbol{v}/(\|\boldsymbol{u}\|^2\|\boldsymbol{v}\|^2)$ as in Example 3.4.

Example 3.6 (Entropic factorization)**.** Assume that the input data is positive. We wish to build a model that imposes a smoothing over the parameters to limit overfitting, and to avoid some of the biases due to the objective function or the optimization method. In such a case, while building an incremental model, we can try to maximize the entropy of the parameters. The resulting problem takes the form

$$\min \quad \Delta(\boldsymbol{T}+\sigma\boldsymbol{u}\boldsymbol{v}^T) + \lambda H(\boldsymbol{u}) + \omega H(\boldsymbol{v}), \tag{3.10}$$

where $H(\boldsymbol{x}) = \sum_i x_i \log x_i - x_i$ is the (unnormalized) negative entropy of $\boldsymbol{x}$. For example, if we wish to maximize the entropy regularized logarithmic cost,

$$\min \; -\sum_{ij} a_{ij}\log(t_{ij}+\sigma u_i v_j) + \lambda H(\boldsymbol{u}) + \omega H(\boldsymbol{v}),$$

we can apply our technique by exploiting the convexity of $-\log x$. Without loss of generality we can restrict $\sum_k \sigma_k = 1$ to avoid degenerate solutions. Then one obtains $\sigma = 1/K$, and the following simple non-linear equations for u_p and v_q

$$u_p \log u_p = (\boldsymbol{M}\mathbf{1})_p/\lambda; \qquad v_q \log v_q = (\boldsymbol{M}^T\mathbf{1})_p/\omega$$

Observe that even though $\sigma = 1/K$, all of the information is actually captured by $\boldsymbol{u}$ and $\boldsymbol{v}$, and σ is merely a scale parameter.

Remark: Observe that if we estimated models of the form $(1-\alpha)\boldsymbol{T}+\alpha(\sigma\boldsymbol{u}\boldsymbol{v}^T)$, then the above log-distortion could be a log-likelihood, and we would obtain an entropic version of an incremental EM algorithm.

Example 3.7 (Incremental KL–I)**.** Another fundamental example considers the KL-Divergence $\Delta(\hat{\boldsymbol{A}}, \boldsymbol{A}) = \mathrm{KL}(\boldsymbol{A}\|\hat{\boldsymbol{A}})$, so that (3.3) takes the form

$$\min_{\sigma,\boldsymbol{u},\boldsymbol{v}} \sum_{ij} \nu_{ij}\left\{a_{ij}\log\frac{a_{ij}}{t_{ij}+\sigma u_i v_j} - a_{ij} + (t_{ij}+\sigma u_i v_j)\right\}.$$

As before we form an auxiliary function for this divergence and differentiate it to obtain the first-order necessary conditions

$$\sum_{ij}\frac{\nu_{ij}a_{ij}\beta_{ij}}{\sigma} = \sum_{ij} u_i v_j, \quad \sum_j \frac{\nu_{pj}a_{pj}\beta_{pj}}{u_p} = \sum_j \sigma v_j,$$
$$\sum_i \frac{\nu_{iq}a_{iq}\beta_{jq}}{v_q} = \sum_i \sigma u_i,$$

which yield the updates

$$\sigma = \frac{\mathbf{1}^T\boldsymbol{M}\mathbf{1}}{\boldsymbol{u}^T\mathbf{1}\boldsymbol{v}^T\mathbf{1}}, \quad u_p = \frac{(\boldsymbol{M}\mathbf{1})_p}{\sigma\boldsymbol{v}^T\mathbf{1}}, \quad v_q = \frac{(\boldsymbol{M}^T\mathbf{1})_q}{\sigma\boldsymbol{u}^T\mathbf{1}}, \tag{3.11}$$

where $\boldsymbol{M} = [\nu_{ij}\beta_{ij}a_{ij}]$. If we normalize $\boldsymbol{u}$ and $\boldsymbol{v}$ to be probability vectors, then (3.11) translate into updates for an incremental version of KL-Divergence NNMA problem.

Example 3.8 (Regularized incremental KL)**.** In the derivation above, we did not use any regularization. With the addition of regularization terms $R_u(\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|^2$, and $R_v(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v}\|^2$, which help to increase the sparsity of the new model $\sigma\boldsymbol{u}\boldsymbol{v}^T$, we obtain the following updates ($\boldsymbol{M}$ is as in Example 3.7)

$$\sigma = \frac{\mathbf{1}^T\boldsymbol{M}\mathbf{1}}{\boldsymbol{u}^T\mathbf{1}\boldsymbol{v}^T\mathbf{1}}, \quad u_p = \frac{(\boldsymbol{M}\mathbf{1})_p}{\sigma\boldsymbol{v}^T\mathbf{1} + \lambda}, \quad v_q = \frac{(\boldsymbol{M}^T\mathbf{1})_q}{\sigma\boldsymbol{u}^T\mathbf{1} + \omega}.$$

Example 3.9 (KL via functional gradients)**.** For simplicity assume that $\sum_{ij} a_{ij} = 1$, so that $\sum_{ij} \hat{a}_{ij} = 1$ also holds. Then, it can be shown that maximizing $\langle \boldsymbol{\nabla}\Delta, \sigma\boldsymbol{u}\boldsymbol{v}^T - \boldsymbol{T}\rangle$ is tantamount to minimizing $\sum_{ij} \frac{\sigma u_i v_j a_{ij}}{t_{ij}}$. With additional quadratic penalty terms, this leads to the incremental regularized aspect model, which was developed in [269] and applied to the task of topic detection and tracking.

Example 3.10 (Incremental KL—II)**.** Assume for now that $R_u, R_v \equiv 0$. We consider minimizing the KL-Divergence, where $\Delta(x, y) = \mathrm{KL}(x\|y) = x\log x - x + y$. Using the fact that $\partial\mathrm{KL}(x\|y)/\partial x = \log(x/y)$, equations (3.7a)–(3.7c) can be simplified after some minor manipulations to yield

$$\begin{aligned} \log\sigma = -\frac{\boldsymbol{u}^T\boldsymbol{M}_1\boldsymbol{v}}{\boldsymbol{u}^T\boldsymbol{N}\boldsymbol{v}}, \quad \log u_p = -\frac{(\boldsymbol{M}_2\boldsymbol{v})_p}{(\boldsymbol{N}\boldsymbol{v})_p}, \\ \log v_q = -\frac{(\boldsymbol{M}_3^T\boldsymbol{u})_q}{(\boldsymbol{N}^T\boldsymbol{u})_q}, \end{aligned} \tag{3.12}$$

where $\boldsymbol{N} = [\nu_{ij}\beta_{ij}]$, $\boldsymbol{M}_1 = [\nu_{ij}\beta_{ij}\log\frac{u_i v_j}{\beta_{ij}a_{ij}}]$, $\boldsymbol{M}_2 = [\nu_{ij}\beta_{ij}\log\frac{\sigma v_j}{\beta_{ij}a_{ij}}]$, and $\boldsymbol{M}_3 = [\nu_{ij}\beta_{ij}\log\frac{\sigma u_i}{\beta_{ij}a_{ij}}]$. These scaled exponentiated updates (3.12) yield an interesting new incremental algorithm for building a latent semantic indexing model.

Example 3.11 (Bregman divergence)**.** Here $\Delta(x, y) = D_\varphi(x; y) = \phi(x) - \phi(y) - \phi'(y)(x - y)$, where ϕ is strictly convex function $\phi : \mathbb{R} \to \mathbb{R}$. These divergences are non-negative, strictly convex in the first argument, and zero iff $x = y$ [45]. They provide a natural generalization of the well known Euclidean distance and KL-Divergence. Using $\partial\Delta(x, y)/\partial x = \phi'(x) - \phi'(y)$, equations (3.7a)–(3.7c) become

$$\sum_{ij} n_{ij}(\delta\phi')_{ij}u_i v_j = 0, \quad \sum_j n_{pj}(\delta\phi')_{pj}\sigma v_j = 0, \quad \sum_i n_{iq}(\delta\phi')_{iq}\sigma u_i = 0, \tag{3.13}$$

where $(\delta\phi')_{ij} = \phi'(\frac{\sigma u_i v_j}{\beta_{ij}}) - \phi'(a_{ij})$, and $n_{ij} = \nu_{ij}\beta_{ij}$. For example, with $\phi(x) = \frac{1}{2}x^2$, and $\phi(x) = x\log x$, Eqn. (3.13) leads to incremental SVD (Example 3.4), and incremental KL (Example 3.10), respectively. With $\phi(x) = -\log x$ we can simplify (3.13) and obtain a procedure for incrementally minimizing the Itakura-Saito distance (burg-divergence), namely

$$\sigma = \frac{\mathbf{1}^T\boldsymbol{M}_1\mathbf{1}}{\boldsymbol{u}^T\boldsymbol{N}\boldsymbol{v}}, \quad u_p = \frac{(\boldsymbol{M}_1\mathbf{1})_p}{\sigma(\boldsymbol{N}\boldsymbol{v})_p}, \quad v_q = \frac{(\boldsymbol{M}_1^T\mathbf{1})_p}{\sigma(\boldsymbol{N}^T\boldsymbol{u})_p}, \tag{3.14}$$

where $\boldsymbol{M}_1 = [\nu_{ij}\beta_{ij}^2]$ and $\boldsymbol{N} = [\nu_{ij}\beta_{ij}/a_{ij}]$. The updates (3.14) bear a close similarity to (3.12), perhaps due to the interesting relation $D_\varphi(x; y)$ with $\phi = -\log x$ equals $\mathrm{KL}(1\|x/y)$.

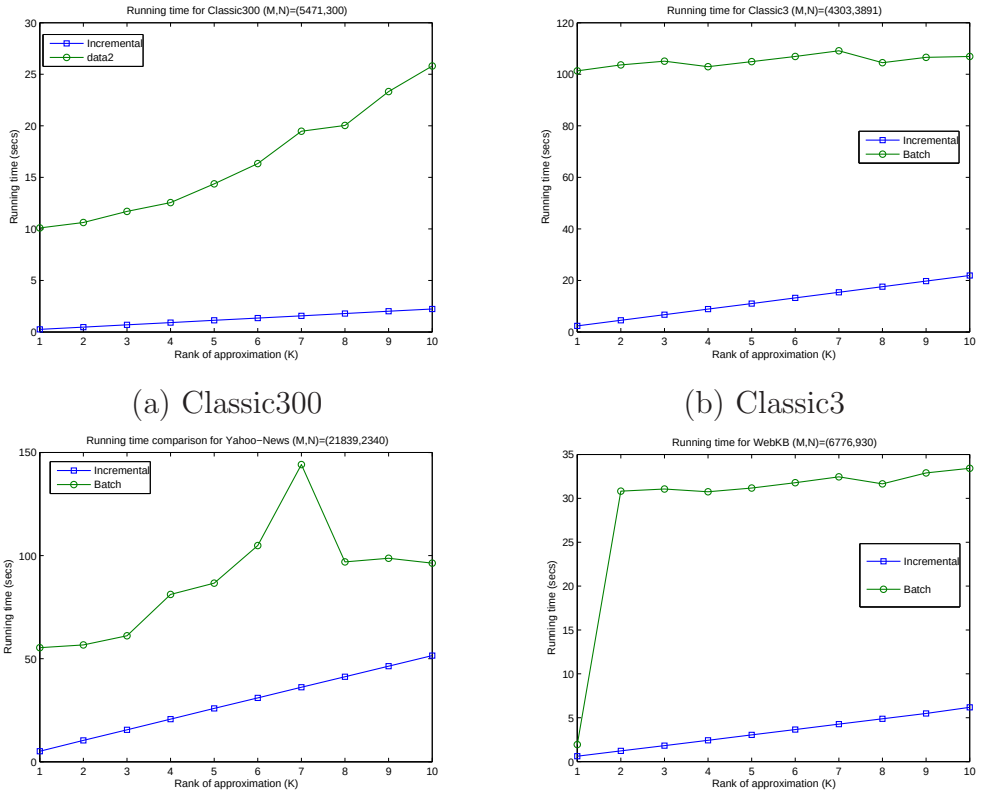
Remarks: As is obvious from the examples above, one can generate countless special cases, each with its own interesting properties. We now look at some experimental results to see how some of the different methods perform in practice.

3.4 Experiments

We present some illustrative experimental results in this section. We remind the reader since many already published methods such as density boosting [239], the vertex direction method [31], incremental aspect models [269], and incremental SVD, are *special* cases of our formulation, their applications and experimental results carry over to this chapter directly. The purpose of this section is to offer some experimental evidence to demonstrate the superiority of our incremental approach. Since the potential number of algorithms is unlimited, we choose to illustrate our point by comparing our incremental low-rank approximation algorithm given in Example 3.7 against the batch algorithm of Lee and Seung [172] that also minimizes a KL-Divergence.

For the purpose of experimentation, we implemented all methods in MATLAB. We remark that since text data is usually large and sparse, our methods are suitable candidates for handling it. The data matrices that we report results on are

1. Classic3: Dataset with 3891 documents drawn from the areas of information retrieval (CISI), aeronautical systems (CRAN), and medical journal



(a) Classic300 (b) Classic3

(c) Yahoo-News (d) WebKB

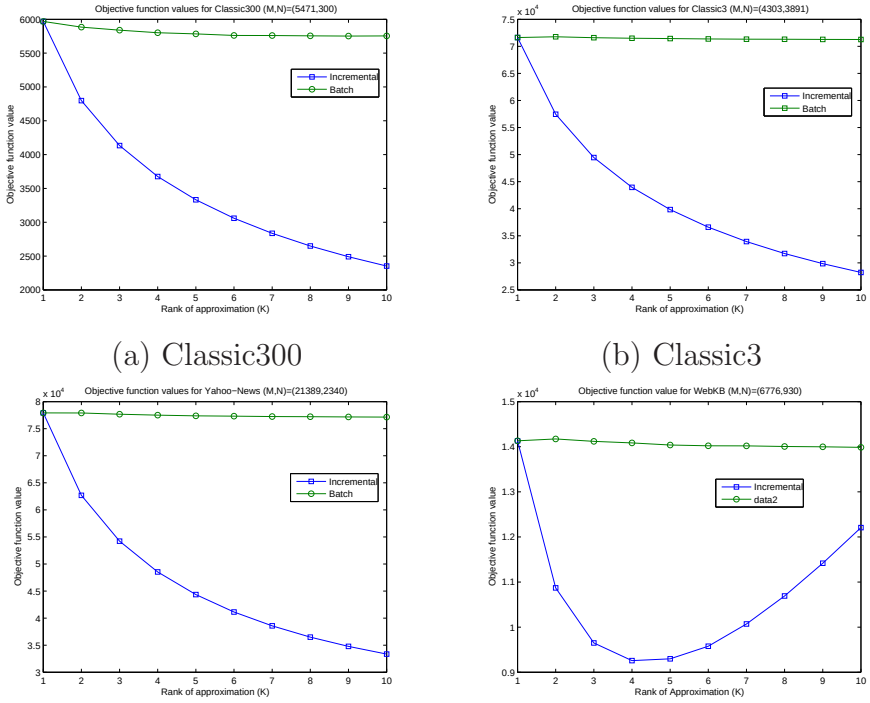
Figure 3.1: Running time on the (a) Classic300, (b) Classic3, (c) Yahoo-News, and (d) WekKB datasets as K is varied. We minimize $\mathrm{KL}(\boldsymbol{A}\|\hat{\boldsymbol{A}})$ using our incremental algorithm. The batch algorithm used was Lee & Seung's KL-Divergence (unnormalized) algorithm [172]. Total time spent by batch method for the four datasets is (164.2s,1051.9s,881.9s,287.9s), while the incremental method takes time (6.85s,36.9s,51.5s,6.2s).

articles (MED). Data matrix was of size 4303×3891.

2. Classic300: A subset of 300 documents taken from Classic3, with 100 randomly chosen documents from each of the three categories given above. Data matrix has size 5471×300.
3. Yahoo News (K-Series): Dataset with 2340 news articles belonging to 20 different categories. The size of this data matrix is 21819×2340.
4. WekKB (Courses subset). Dataset with 930 webpages about courses. The size of the data matrix is 6776×930.

Figure 3.1(a) reports running time comparisons on the Classic300 dataset when obtaining a KL-Divergence based low-rank approximation. The timing experiments reported are indicative and not absolute, since both algorithms could benefit from better implementations. From the figure, the benefit of incrementality is evident, whereby one observes a large savings in time. The total time the batch algorithm needed for producing the values on the plot is the sum of its time taken for each value of K, because the batch approach needs to be run afresh with changing K. On the other hand, our incremental approach needs to be run just once with $K = 10$, producing all the other values as intermediate steps.

Somewhat surprisingly the incremental method achieves better objective function values than the batch algorithm (see Figure 3.2), and that too, at a fraction of the running time. This figure offers encouragement for the potential of our incremental approach. However, we do remark that the batch



(a) Classic300 (b) Classic3

(c) Yahoo-News (d) WebKB

Figure 3.2: Objective function values on the (a) Classic300, (b) Classic3 (b), (c) Yahoo-News, and (d) WebKB datasets as K is varied. The experimental setting is the same as in Figure 3.1.

algorithm gives a better *subjective* coverage of the underlying categories in the dataset, than the incremental algorithm. Investigating this behavior is currently underway, and it involves solving the hard problem of determining which objective function to use for a given problem.

3.5 Extensions

In this chapter we have so far used strictly convex regularization functions R_u and R_v. However, there are many interesting problems that can benefit from having strictly concave penalties. Consider, for example, the approximation problem

$$\min \Delta(\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \boldsymbol{A}) - \lambda R_U(\boldsymbol{U}) - \omega R_V(\boldsymbol{V}), \tag{3.15}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ can be estimated either in batch mode or incrementally. Problem (3.15) typically arises when a min-max type concern is at play. More specifically, consider an incremental approximation of (3.15) that aims to make every new incoming model $\sigma\boldsymbol{u}\boldsymbol{v}^T$ as distinct as possible from the previously estimated parameters $\boldsymbol{T}$. This requirement is not artificial—in a topic discovery system, often many incremental methods end up rediscovering previously discovered topics. To avoid such a scenario, one approach is to maximize the distance of $\boldsymbol{u}$ from the previously computed $\boldsymbol{u}_k$ values (similarly for $\boldsymbol{v}$). To that end one can select $R_u(\boldsymbol{u}) = \|\boldsymbol{u} - \sum_{k=1}^{K-1} \boldsymbol{u}_k\|^2$, or some other appropriate penalty. One could use the auxiliary function technique of Section 3.2.1, or exploiting the fact that Δ is convex, and R_U, R_V are concave, one could use the concave-convex procedure [294] (which itself is essentially an auxiliary function technique).

3.5.1 Future work

Another exciting extension of this work is to handle truly incremental or online scenarios—i.e., adapting the approximation as new arrives. A simple first step method is to project the new data onto the subspace spanned by current low-rank model. If the error of such a projection is large, then we can run the incremental phase to update the approximation, or to discard previously computed models (for e.g., to admit "forgetting").

A more challenging aspect comes from an implementation viewpoint. There exists an inherent trade-off between batch and incremental approaches. For example, batch algorithms can be usually implemented using blocked (BLAS3 and BLAS2) computations, while the incremental algorithms usually have at most matrix-vector (BLAS2) operations. Instead of updating the incremental approximation one rank at a time, one can update an entire "block", wherein we proceed in chunks of say B blocks at a time. Such an approach has precedent in the Block Lanczos procedure [106] for computing the SVD. The challenge would be to produce special purpose incremental algorithms that automatically trade-off between blocked and unblocked updates.

Chapter 4

Metric Nearness

Note: The material within this chapter is based on [39, 86].

4.1 Introduction

In this chapter we depart from the structured low-rank approximation studied in Chapters 2 and 3 to study a matrix nearness problem of the type (1.2). We will take up low-rank approximation once again in Chapter 5, after we have developed some important algorithmic techniques in this chapter. The focus of this chapter is a problem called *Metric Nearness*, which refers to the problem of optimally restoring metric properties to distance measurements that happen to be non-metric due to measurement errors or otherwise. This problem was briefly introduced in Section 1.2.1 and we take up its detailed study in this chapter.

4.1.1 Background and Motivation

Most applications make some assumptions about the properties that the input data should satisfy. Due to measurement errors, noise, or an inability to gather data completely, an application may receive data that do not

conform to its requirements. For example, imagine taking measurements as a part of some experiment. The theory suggests that the quantities measured should represent distances between points in a discrete metric space. However, measurements being what they are, one ends up with a set of numbers that do not represent actual distance values, primarily because they fail to satisfy the triangle inequality. It might be beneficial to somehow optimally massage the measurements to obtain a set of "nearest" distance values that obey the properties of a metric. It could also happen that experimental expenses and difficulties prevent one from making all the measurements. Before this incomplete set of measurements can be used in an application it might need to be tweaked, preferably minimally. As before, obtaining a "nearest" set of distance values (measurements) seems to be desirable.

Both scenarios above lead to the *metric nearness problem:* Given a set of input distances, find a "nearest" set of output distances that satisfy the properties of a metric. The notion of nearness is quantified by the function that measures distortion between the input and output distances.

If there are N points, we may collect the measurements into an $N \times N$ symmetric matrix whose (j, k) entry represents the distance (or some measure of proximity) between points j and k. Then, we seek to approximate this matrix by another (say $\boldsymbol{M}$) whose entries satisfy the triangle inequalities. That is, $m_{ij} \leq m_{ik} + m_{kj}$ for every triple (i, k, j). Any such matrix will represent the distances among N points in some metric space. Note that in comparison to the previous chapter, we do note denote points by $\boldsymbol{a}_i$, and we

just refer to the i-th point. This distinction arises due to the fact that for the metric nearness problem, the actual points are immaterial and only interpoint distances are what we care about.

While approximating the input matrix, we calculate the error with a distortion measure that depends on how the corrected matrix should relate to the input matrix. For example, one might prefer to change a few entries significantly or to change all the entries a little. We will focus on using vector norms for characterizing the error of approximation, with a brief extension to more general distortion measures such as Bregman divergences (introduced in Chapter 2).

There is no analytic solution to the metric nearness problem. Fortunately this problem lends itself to a convex formulation, whereby developing algorithms for solving it becomes much easier. However, despite the natural convexity of the formulations, the large number of triangle inequality constraints can make traditional approaches or general purpose convex programming software overly slow. Therefore, to efficiently solve the metric nearness problem one needs to exploit its inherent structure; we develop these ideas further in this chapter.

4.2 Problem Formulation

We begin with two primary definitions.

Definition 4.1 (Dissimilarity matrix)**.** A matrix $\boldsymbol{D}$ is called a dissimilarity

matrix if it is symmetric and nonnegative, and has a zero diagonal.

Dissimilarity matrices are used to represent pairwise proximity data between objects in an application.

Definition 4.2 (Distance matrix)**.** A matrix $\boldsymbol{M}$ is called a distance matrix if it is a dissimilarity matrix whose entries satisfy the triangle inequalities. Specifically, $\boldsymbol{M}$ is a distance matrix if

$$\begin{aligned} & m_{ij} \geq 0, && m_{ii} = 0, \quad m_{ij} = m_{ji}, \\ \text{and} \quad & m_{ij} \leq m_{ik} + m_{kj} && \text{for distinct triples } (i, k, j). \end{aligned}$$

We remark that symmetry, while part of the definition of a metric, is not crucial to our algorithms; asymmetry can be handled at the expense of doubling the running time and storage.

It is easily observed that distance matrices contain $\binom{N}{2}$ parameters, and that the set of all $N \times N$ distance matrices forms a closed convex polyhedral cone, which we denote as $\mathscr{M}_N$. Further note that for without loss of generality we drop the requirement that $m_{ij} \neq 0$ unless $i = j$, thereby actually considering pseudo-metrics.

Assume that the input is a dissimilarity matrix $\boldsymbol{D}$. Metric nearness seeks a distance matrix $\boldsymbol{M}$ that is closest to $\boldsymbol{D}$, with respect to some measure of "closeness." Formally, we seek a matrix $\boldsymbol{M}$ so that

$$\boldsymbol{M} \in \operatorname*{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \Delta(\boldsymbol{X}, \boldsymbol{D}), \tag{4.1}$$

where Δ is a distortion function as usual. For computational ease, we restrict our attention to the the case where Δ is a convex function of $\boldsymbol{X}$. For the most part, we will limit our discussion to the *vector* ℓ_p norms, wherein we treat the strict upper triangular part of our matrices as vectors.

4.2.1 Metric Nearness for the ℓ_2 norm

We start with a formulation for the vector ℓ_2 norm based metric nearness problem. Given the input dissimilarity matrix $\boldsymbol{D} = [d_{ij}]$ (where $d_{ij} = d_{ji}$), the problem (4.1) becomes

$$\min_{\boldsymbol{X} \in \mathscr{M}_N} \quad \frac{1}{2} \sum_{i<j} (x_{ij} - d_{ij})^2.$$

Note that the sum above ranges over $i < j$, since the involved matrices are symmetric and have a zero diagonal.

Let T_N be the set of $3\binom{N}{3}$ triples, each of which corresponds to a triangle inequality that the entries of an $N \times N$ distance matrix must satisfy. Formally,

$$T_N = \{(i,j,k), (j,k,i), (k,i,j) : 1 \leq i < k < j \leq N\}, \tag{4.2}$$

where the triple (i,k,j) corresponds to the triangle inequality

$$x_{ij} \leq x_{ik} + x_{kj}.$$

With the introduction of an auxiliary matrix $\boldsymbol{E} = \boldsymbol{X} - \boldsymbol{D}$ that represents the changes to the original dissimilarities, the ℓ_2 metric nearness problem can be

rewritten as the following quadratic program:

$$\underset{e_{ij}}{\text{Minimize}}\ \frac{1}{2}\sum_{i<j} e_{ij}^2, \tag{4.3}$$

$$\text{subject to}\ \ e_{ij} - e_{ik} - e_{kj} \ \leq\ \ d_{ik} + d_{kj} - d_{ij} = v_{ikj} \quad \forall\ (i,k,j) \in T_N. \tag{4.4}$$

The triangle inequality constraints are encoded by (4.4). Since the ℓ_2 norm is strictly convex, the solution to (4.3) is unique. The variable v_{ikj} quantifies the *violation* in the (i,k,j) triangle inequality.

4.2.2 Metric Nearness for the ℓ_1 and ℓ_∞ norms

When measuring approximation error using the ℓ_1 norm, we wish to minimize

$$\sum_{i<j} |e_{ij}|, \tag{4.5}$$

where $e_{ij} = x_{ij} - d_{ij}$ as in the previous section. However, to write the problem as a linear program, we need to introduce additional variables $f_{ij} = |e_{ij}|$. The resulting problem is the following linear program:

$$\underset{e_{ij}, f_{ij}}{\text{Minimize}}\ \ \sum_{i<j} \big(1 \cdot f_{ij} + 0 \cdot e_{ij}\big), \tag{4.6}$$

$$\begin{array}{rrcll} & e_{ij} - e_{ik} - e_{kj} & \leq & v_{ikj} & \forall\ (i,k,j) \in T_N, \\ \text{subject to} & -e_{ij} - f_{ij} & \leq & 0 & 1 \leq\ i < j\ \leq N, \\ & e_{ij} - f_{ij} & \leq & 0 & 1 \leq\ i < j\ \leq N. \end{array} \tag{4.7}$$

The fact that $f_{ij} = |e_{ij}|$ is accomplished by the latter two sets of inequalities in (4.7).

Similarly, for the ℓ_∞ nearness problem, we introduce a variable $\zeta = \max_{ij} |e_{ij}|$ that represents the vector ℓ_∞ norm of $\boldsymbol{E}$. The ℓ_∞ nearness problem

becomes:

$$\underset{e_{ij},\zeta}{\text{Minimize}} \quad \zeta + \sum_{i<j} 0 \cdot e_{ij}, \tag{4.8}$$

$$\begin{array}{lrcll} & e_{ij} - e_{ik} - e_{kj} & \leq & v_{ikj} & \forall\ (i,k,j) \in T_N, \\ \text{subject to} & -e_{ij} - \zeta & \leq & 0 & 1 \leq\ i < j\ \leq N, \\ & e_{ij} - \zeta & \leq & 0 & 1 \leq\ i < j\ \leq N. \end{array} \tag{4.9}$$

The latter two sets of inequalities in (4.9) express the fact $|e_{ij}| \leq \zeta$ for all i and j.

4.2.3 Metric nearness for ℓ_p norms

Metric nearness may be easily formulated for ℓ_p norms, where $1 < p < \infty$. The problem is the following convex program:

$$\begin{array}{ll} \underset{e_{ij}}{\text{Minimize}} & \dfrac{1}{p} \displaystyle\sum_{i<j} |e_{ij}|^p, \\ \text{subject to} & e_{ij} - e_{ik} - e_{kj} \ \leq \ \ v_{ikj} \quad \forall\ (i,k,j) \in T_n. \end{array}$$

Since the ℓ_p norms are strictly convex for $1 < p < \infty$, the associated metric nearness problems have unique solutions. There is a basic intuition for choosing p when solving the nearness problems. The ℓ_1 norm error is computed as the absolute sum of changes to the input matrix, while ℓ_∞ reflects only the maximum absolute change. The other ℓ_p norms interpolate between these two extremes. Thus, a small value of p typically results in a solution that prefers a few large changes to the original data, while a large p typically results in a solution with many small changes. In practice, however, the ℓ_1, ℓ_2, and ℓ_∞ problems are computationally easier to solve than those using arbitrary ℓ_p norms. Thus, we focus primarily on these three problems.

4.2.4 Metric nearness for KL-Divergence

All the formulations of metric nearness above were for vector norms. Now we look at the special case where the distortion is measured using a KL-Divergence, particularly because it leads to multiplicative updates. The corresponding optimization problem is

$$\begin{aligned} \min \quad & \sum_{ij} x_{ij} \log \frac{x_{ij}}{d_{ij}} - x_{ij} + d_{ij} \\ \text{subject to} \quad & x_{ij} \leq x_{ik} + x_{kj} \quad \forall (i,k,j) \in T_N. \end{aligned} \tag{4.10}$$

This problem has the important characteristic, that if $d_{ij} = 0$, then the corresponding x_{ij} must be zero, else the divergence is undefined. More generally, it will tend to more strongly keep small distances small.

4.3 Triangle Fixing Algorithms

The previous section formulated the metric nearness problem as a quadratic program for the ℓ_2 norm, as a linear program for ℓ_1 and ℓ_∞ norms, and as a convex program for ℓ_p norms and the KL-Divergence. Using off-the-shelf software for these formulations might appear to be an attractive way to solve the corresponding problems. However, it turns out that the computational time and storage requirements of such an approach can be prohibitive, especially due to the $O(N^3)$ number of inequality constraints. An efficient algorithm must exploit the inherent structure offered by these triangle inequality constraints. In this section, we develop *triangle fixing algorithms*, which take advantage of this structure to efficiently solve the problem. These algorithms

iterate through the triangle inequalities, optimally enforcing any inequality that is not satisfied. While enforcing the triangle inequalities, one needs to introduce appropriate correction terms to guide the iterative algorithm to the globally optimal solution. The details are provided below.

4.3.1 Triangle fixing for ℓ_2 metric nearness

Our approach for solving (4.3) is iterative, and is based on the technique described in [45]. Collecting all the e_{ij} values into vector $\boldsymbol{e}$ and the violation amounts into $\boldsymbol{v}$, problem (4.3) may be rewritten as

$$\begin{aligned} \min_{\boldsymbol{e}} \quad & \tfrac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{e} \leq \boldsymbol{v}, \end{aligned} \tag{4.11}$$

where matrix $\boldsymbol{A}$ encodes the triangle inequalities (4.4), whereby, each row of $\boldsymbol{A}$ has one $+1$ entry, and two -1 entries. This matrix has some interesting properties, for e.g., (i) each column of $\boldsymbol{A}$ has just $3(N-2)$ entries, (ii) $\boldsymbol{A}$ has full column rank, and (iii) $\boldsymbol{A}$ has only three distinct singular values, viz.,$\sqrt{3N-4}$, $\sqrt{2N-2}$ and $\sqrt{N-2}$ with multiplicities $\frac{1}{2}N(N-3)$, $(N-1)$, and 1, respectively.

Now we proceed to solve (4.11). The Lagrangian of (4.11) is

$$L(\boldsymbol{e}, \boldsymbol{z}) = \tfrac{1}{2}\boldsymbol{e}^T\boldsymbol{e} + \langle \boldsymbol{z},\, \boldsymbol{A}\boldsymbol{e} - \boldsymbol{v} \rangle,$$

where $\boldsymbol{z}$ is the dual vector. A necessary condition for optimality of (4.11) is

$$\frac{\partial L}{\partial \boldsymbol{e}} = 0 \quad \Longrightarrow \quad \boldsymbol{e} = -\boldsymbol{A}^T\boldsymbol{z}, \quad \boldsymbol{z} \geq \boldsymbol{0}. \tag{4.12}$$

We start the iterative minimization procedure by initializing both $\boldsymbol{e}$ and $\boldsymbol{z}$ to zero as this choice satisfies (4.12). At each subsequent iterative step, we add a "correction" to the dual variable $\boldsymbol{z}$ that ensures that (4.12) is maintained, and the associated dual function $g(\boldsymbol{z}) = \min_{\boldsymbol{e}} L(\boldsymbol{e}, \boldsymbol{z})$ is nondecreasing. Let $\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{\theta}$ be the dual vector after "correction"; then, to maintain (4.12) we must update $\boldsymbol{e}$ as follows

$$\boldsymbol{e}' = -\boldsymbol{A}^T \boldsymbol{z}' = -\boldsymbol{A}^T(\boldsymbol{z} + \boldsymbol{\theta}) = \boldsymbol{e} - \boldsymbol{A}^T\boldsymbol{\theta}.$$

The simplest correction entails updating the dual vector $\boldsymbol{z}$ one coordinate at a time. Such an update corresponds to "fixing" (enforcing) one triangle inequality at a time, hence the name of our procedure. Assume that the dual variable corresponding to inequality (i, k, j) is updated, i.e., $z'_{ikj} = z_{ikj} + \theta$. Then, to respect (4.12) we must make the update $\boldsymbol{e}' = \boldsymbol{e} - \theta \boldsymbol{a}_{ikj}$, where $\boldsymbol{a}_{ikj}$ is the row of matrix $\boldsymbol{A}$ associated with the (i, k, j) inequality. Recall that $\boldsymbol{a}_{ikj}$ has only three non-zero entries corresponding to the edges (i, j), (i, k) and (k, j). Hence, exactly three entries in $\boldsymbol{e}$ need to be updated to form $\boldsymbol{e}'$. We make an ℓ_2-minimal change to $\boldsymbol{e}$ while enforcing the (i, k, j) inequality, to obtain $\boldsymbol{e}'$. That is, we solve

$$\begin{aligned} \min_{\boldsymbol{e}'} \quad & \|\boldsymbol{e}' - \boldsymbol{e}\|_2^2 \\ \text{subject to} \quad & \langle \boldsymbol{a}_{ikj},\, \boldsymbol{e}' \rangle = v_{ikj}. \end{aligned} \tag{4.13}$$

The solution to (4.13) may be obtained by computing the parameter μ, so that

$$\boldsymbol{e}' = \boldsymbol{e} - \mu \boldsymbol{a}_{ikj} \quad \text{and} \quad \langle \boldsymbol{a}_{ikj},\, \boldsymbol{e}' \rangle = v_{ikj}.$$

ALGORITHM 4.2: Metric Nearness for ℓ_2 norm.

METRIC_NEARNESS_L2($\boldsymbol{D}$, κ)
Input: Dissimilarity matrix $\boldsymbol{D}$, tolerance κ
Output: $\boldsymbol{M} = \text{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \|\boldsymbol{X} - \boldsymbol{D}\|_2$.
{Initialize the primal and the dual variables}
$e_{ij} \leftarrow 0$ for $1 \leq i < j \leq N$
$(z_{ijk}, z_{jki}, z_{kij}) \leftarrow 0$ for $1 \leq i < k < j \leq N$
$\delta \leftarrow 1 + \kappa$
while $(\delta > \kappa)$ {convergence test}
  foreach triangle inequality (i, k, j)
    $v \leftarrow d_{ik} + d_{kj} - d_{ij}$ {Compute violation}
    $\mu \leftarrow \frac{1}{3}(e_{ij} - e_{ik} - e_{kj} - v)$ $(\star)$
    $\theta \leftarrow \max\{\mu, -z_{ikj}\}$ {Stay within half-space of constraint}
    $e_{ij} \leftarrow e_{ij} - \theta$, $e_{ik} \leftarrow e_{ik} + \theta$, $e_{kj} \leftarrow e_{kj} + \theta$ $(\star\star)$
    $z_{ikj} \leftarrow z_{ikj} + \theta$ {Add correction to dual variable}
  end *foreach*
  $\delta \leftarrow$ sum of changes in the e_{ij} values
end *while*
return $\boldsymbol{M} = \boldsymbol{D} + \boldsymbol{E}$

It is easy to verify that μ is given by

$$\mu = \frac{1}{\|\boldsymbol{a}_{ikj}\|^2}(\boldsymbol{a}_{ikj}^T \boldsymbol{e} - v_{ikj}) = \frac{1}{3}(\boldsymbol{a}_{ikj}^T \boldsymbol{e} - v_{ikj}).$$

We could now let $\theta = \mu$, so that $z'_{ikj} = z_{ikj} + \mu$, and $\boldsymbol{e}' = \boldsymbol{e} - \mu \boldsymbol{a}_{ikj}$. However, we also need to ensure that $z'_{ikj} \geq 0$. Thus, we let $z'_{ikj} = z_{ikj} + \theta$, and $\boldsymbol{e}' = \boldsymbol{e} - \theta \boldsymbol{a}_{ikj}$ instead, where $\theta = \max\{\mu, -z_{ikj}\}$. Algorithm 4.2 puts together all these ideas to give the complete iterative triangle fixing procedure.

Remarks The procedure derived above ensures that at each iteration $g(\boldsymbol{z}') \geq g(\boldsymbol{z})$, i.e., it is a *dual coordinate ascent* procedure. Following [45], it can be shown that in the limit, the $\boldsymbol{A}\boldsymbol{e} \leq \boldsymbol{v}$ constraints are satisfied. Since (4.12) is

also maintained at each step, the KKT conditions, which are necessary and sufficient for this problem, are satisfied in the limit. Thus, the triangle fixing procedure converges to the optimal solution of (4.11). In fact, Algorithm 4.2 is an efficient version of Bregman's method for minimizing a convex function subject to linear inequality constraints [45]. Our algorithm exploits the structure of the problem to obtain its efficiency.

4.3.2 Triangle fixing for ℓ_1 and ℓ_∞

Triangle fixing is somewhat less direct for the ℓ_1 and ℓ_∞ problems. The reason these norms pose an additional challenge is because they are not strictly convex; the convergence of the basic triangle fixing procedure depends on the strict convexity of the norm used. We illustrate only the ℓ_1 case; the development for ℓ_∞ takes the same course.

With the introduction of vector and matrix notation, the ℓ_1 matrix nearness problem may be rewritten as

$$\begin{aligned} &\min_{\boldsymbol{e},\boldsymbol{f}} \ \mathbf{0}^T\boldsymbol{e} + \mathbf{1}^T\boldsymbol{f}, \\ &\text{subject to } \ \boldsymbol{A}\boldsymbol{e} \le \boldsymbol{v}, \quad -\boldsymbol{e} - \boldsymbol{f} \le \mathbf{0}, \quad \boldsymbol{e} - \boldsymbol{f} \le \mathbf{0}. \end{aligned} \tag{4.14}$$

The auxiliary variable $\boldsymbol{f}$ is interpreted as the element-wise absolute value of $\boldsymbol{e}$. The violations to the triangle inequalities are again given by the vector $\boldsymbol{v}$.

To solve the linear program (4.14) without sacrificing the advantages of triangle fixing we replace it with an equivalent quadratic program. This replacement hinges upon a connection between linear and quadratic programs

that may be motivated by the observation,

$$\operatorname{argmin}_{\boldsymbol{g}} \|\boldsymbol{g}+\epsilon^{-1}\boldsymbol{c}\|^2 \quad = \quad \operatorname{argmin}_{\boldsymbol{g}} (\boldsymbol{g}^T\boldsymbol{g}+2\epsilon^{-1}\boldsymbol{g}^T\boldsymbol{c}+\epsilon^{-2}\boldsymbol{c}^T\boldsymbol{c}) \quad \approx \quad \operatorname{argmin}_{\boldsymbol{g}} \boldsymbol{g}^T\boldsymbol{c},$$

if ϵ is chosen to be sufficiently small (so that the $2\epsilon^{-1}\boldsymbol{g}^T\boldsymbol{c}$ term dominates the objective function). The following theorem, which follows from a result of [185, Theorem 2.1-a-i], makes the above connection concrete.

Theorem 4.3 (ℓ_1 Metric Nearness)**.** *Let* $\boldsymbol{g} = [\boldsymbol{e}; \boldsymbol{f}]$ *and* $\boldsymbol{c} = [\boldsymbol{0}; \boldsymbol{1}]$ *be partitioned conformally. If* (4.14) *has a solution, then there exists an* $\epsilon_0 > 0$*, such that for all* $\epsilon \leq \epsilon_0$*,*

$$\operatorname*{argmin}_{\boldsymbol{g}\in G} \|\boldsymbol{g} + \epsilon^{-1}\boldsymbol{c}\|_2 \quad = \quad \operatorname*{argmin}_{\boldsymbol{g}\in G^\star} \|\boldsymbol{g}\|_2, \tag{4.15}$$

where G *is the feasible set for* (4.14) *and* $G^\star$ *is the set of optimal solutions to* (4.14)*. The minimizer of* (4.15) *is unique.*

From (4.14) one can see that the triangle inequality constraints involve only $\boldsymbol{e}$ and not $\boldsymbol{f}$. This circumstance permits us to use triangle fixing once again. As before, we go through the constraints one by one. The first $3\binom{N}{3}$ constraints are triangle constraints and are handled by triangle fixing. The remaining $2\binom{N}{2}$ absolute value constraints are very simple and thus are enforced easily.

For the ℓ_2 case, the dual variables (corresponding to each constraint) were represented by the vector $\boldsymbol{z}$. For (4.15), we let the dual variables be $[\boldsymbol{z};\ \boldsymbol{\lambda};\ \boldsymbol{\mu}]$; vector $\boldsymbol{z}$ corresponds to the triangle inequalities, while vectors $\boldsymbol{\lambda}$

and $\boldsymbol{\mu}$ correspond to $-\boldsymbol{e} - \boldsymbol{f} \leq \mathbf{0}$ and $\boldsymbol{e} - \boldsymbol{f} \leq \mathbf{0}$, respectively. Together, non-negative values of $\boldsymbol{z}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ correspond to the feasible set G alluded to by Theorem 4.3.

Our augmented triangle-fixing procedure is as follows. First we initialize $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{z}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ so that the first order optimality conditions derived from (4.15) are initially true. Thereafter, we enforce constraints one by one to ensure the the dual functional corresponding to (4.15) is increasing and first order optimality conditions are maintained. Written out as Algorithm 4.3, this procedure becomes an efficient adaptation of Bregman's method, thereby, after a sufficient number of iterations, it converges to the globally optimal solution.

ALGORITHM 4.3: Metric nearness for ℓ_1 norm.

L1_METRIC_NEARNESS($\boldsymbol{D}$, ϵ, κ)
Input: Dissimilarity matrix $\boldsymbol{D}$; tolerance κ; ℓ_1 parameter ϵ
Output: $\boldsymbol{M} \in \operatorname{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \|\boldsymbol{X} - \boldsymbol{D}\|_1$
{Initialization}
$e_{ij} \leftarrow 0$; $f_{ij} = -\epsilon^{-1}$ for $1 \leq i < j \leq N$ {Primal variables}
$(z_{ijk}, z_{jki}, z_{kij}) \leftarrow 0$ for $1 \leq i < k < j \leq N$ {Dual variables – triangles}
$\lambda_{ij} \leftarrow \pi_{ij} \leftarrow 0$ for $1 \leq i < j \leq N$ {Dual variables – Other}
$\delta \leftarrow 1 + \kappa$
while $(\delta > \kappa)$ {convergence test}
  Do triangle fixing on the e_{ij} as in Algorithm 4.2
  {Enforce $-\boldsymbol{e} - \boldsymbol{f} \leq \boldsymbol{0}$ and $\boldsymbol{e} - \boldsymbol{f} \leq \boldsymbol{0}$ as follows}
  $\boldsymbol{\mu} \leftarrow \frac{1}{2}(\boldsymbol{e} + \boldsymbol{f})$ {Projection parameters}
  $\boldsymbol{\theta} \leftarrow \min\{\boldsymbol{\mu}, \boldsymbol{\lambda}\}$ {Correction amount}
  $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \boldsymbol{\theta}$ {Correct dual vector corr. to $-\boldsymbol{e} - \boldsymbol{f} \leq \boldsymbol{0}$}
  $\boldsymbol{e} \leftarrow \boldsymbol{e} - \boldsymbol{\theta}$; $\boldsymbol{f} \leftarrow \boldsymbol{f} - \boldsymbol{\theta}$ {Update primal variables}
  $\boldsymbol{\nu} \leftarrow \frac{1}{2}(\boldsymbol{f} - \boldsymbol{e})$
  $\boldsymbol{\theta} \leftarrow \min\{\boldsymbol{\nu}, \boldsymbol{\pi}\}$ {Correction amount}
  $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} - \boldsymbol{\theta}$ {Correct dual vector corr. to $\boldsymbol{e} - \boldsymbol{f} \leq \boldsymbol{0}$}
  $\boldsymbol{e} \leftarrow \boldsymbol{e} + \boldsymbol{\theta}$; $\boldsymbol{f} \leftarrow \boldsymbol{f} - \boldsymbol{\theta}$ {Update primal variables}
  {Update convergence test parameter}
  $\delta \leftarrow$ sum of absolute changes in e_{ij}.
end.
return $M = \boldsymbol{D} + \boldsymbol{E}$.

Remarks Algorithm 4.3 depends on the parameter ϵ that governs convergence to the true optimal solution. Some care must be exercised in selecting this parameter. After experimentation with random dissimilarity matrices we found that setting $\epsilon^{-1} \approx \max_{ij} d_{ij}$, worked well for Algorithm 4.3. In practice, ϵ should be selected after experimentation, and we are not aware of a theoretically sound way of picking ϵ.

4.3.3 Triangle fixing for other ℓ_p norms

We can go a step further and extend triangle fixing to solve the metric nearness problem for all ℓ_p ($1 < p < \infty$) norms. The problem may be compactly stated as

$$\min_{\boldsymbol{e}} \ \frac{1}{p} \|\boldsymbol{e}\|_p^p \qquad \text{subject to} \qquad \boldsymbol{A}\boldsymbol{e} \leq \boldsymbol{v}. \tag{4.16}$$

Recall that for ℓ_2 metric nearness, at each iterative step we obtained $\boldsymbol{e}'$ from $\boldsymbol{e}$ orthogonally projecting $\boldsymbol{e}$ onto the hyperplane defined by $\langle \boldsymbol{a}_{ikj}, \boldsymbol{e}' \rangle = v_{ikj}$. For the ℓ_p norm problem, we must instead perform a generalized projection, called a *Bregman projection*, which involves solving the following problem (*cf.* 4.13)

$$\min_{\boldsymbol{e}'} \ \varphi(\boldsymbol{e}') - \varphi(\boldsymbol{e}) - \langle \nabla\varphi(\boldsymbol{e}), \boldsymbol{e}' - \boldsymbol{e} \rangle \quad \text{such that } \langle \boldsymbol{a}_{ikj}, \boldsymbol{e}' \rangle = \boldsymbol{v}_{ikj}, \tag{4.17}$$

where $\varphi(\boldsymbol{x}) = \frac{1}{p} \|\boldsymbol{x}\|_p^p$. We use $(\nabla\varphi(\boldsymbol{x}))_i = \operatorname{sgn}(x_i)\, |x_i|^{p-1}$ to determine the projection (4.17) by solving

$$\nabla\varphi(\boldsymbol{e}') = \nabla\varphi(\boldsymbol{e}) + \mu \boldsymbol{a}_{ikj} \qquad \text{so that} \qquad \langle \boldsymbol{a}_{ikj}, \boldsymbol{e}' \rangle = v_{ikj}. \tag{4.18}$$

Since $\boldsymbol{a}_{ikj}$ has only three nonzero entries, once again $\boldsymbol{e}$ needs to be updated in only three components. Therefore, in Algorithm 4.2 we may replace ($\star$) by an appropriate numerical computation of the parameter μ, and replace ($\star\star$) by the computation of the new value of $\boldsymbol{e}$ as resulting from (4.18). As before, each iteration maintains the necessary condition $\partial L(\boldsymbol{e}, \boldsymbol{z})/\partial \boldsymbol{e} = 0$ while correcting the dual vector $\boldsymbol{z}$, and the overall algorithm converges to the optimum of (4.16).

4.3.4 KL Divergence Metric Nearness

As an final variation of Metric Nearness, we illustrate the derivation of an iterative metric nearness algorithm for KL-Divergence between the distance matrix $\boldsymbol{X}$ and the input $\boldsymbol{D}$. The iterative solution involves multiplicative (as opposed to additive) updates to the d_{ij} values when computing the metric solution.

ALGORITHM 4.4: Metric Nearness for KL-Divergence.

METRIC_NEARNESS_KL($\boldsymbol{D}$, κ)
Input: Dissimilarity matrix $\boldsymbol{D}$, tolerance κ
Output: $\boldsymbol{M} = \text{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \text{KL}(\boldsymbol{X}, \boldsymbol{D})$.
{Initialize the primal and the dual variables}
$x_{ij} \leftarrow\ d_{ij}$ for $1 \leq i < j \leq N$
$(z_{ikj}, z_{jki}, z_{kij}) \leftarrow\ 0$ for $1 \leq i < j < k \leq N$
$\delta \leftarrow 1 + \kappa$
while $(\delta > \kappa)$ {convergence test}
  foreach triangle inequality (i, k, j)
    $\mu \leftarrow \frac{1}{2} \log \frac{x_{kj}+x_{ji}}{x_{ik}}$ $(\star)$
    $\theta \leftarrow \min\{\mu, z_{ikj}\}$ {Stay within half-space of constraint}
    $y \leftarrow e^{\theta}$ {Scaling factor}
    $x_{ik} \leftarrow x_{ik}y$, $x_{kj} \leftarrow x_{kj}/y$, $x_{ji} \leftarrow x_{ji}/y$ $(\star\star)$
    $z_{ikj} \leftarrow z_{ikj} - \theta$ {Add correction to dual variable}
  end *foreach*
  $\delta \leftarrow$ sum of changes in the x_{ij} values
end *while*
return $\boldsymbol{M} = \boldsymbol{X}$

As before, in vector notation the problem may be written as

$$\begin{aligned} &\min_{\boldsymbol{x}} \text{KL}(\boldsymbol{x}, \boldsymbol{d}) \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{0}, \end{aligned} \tag{4.19}$$

where

$$\mathrm{KL}(\boldsymbol{x}, \boldsymbol{d}) = \sum_{i<j} x_{ij} \log \frac{x_{ij}}{d_{ij}} - x_{ij} + d_{ij}.$$

Letting $\varphi(\boldsymbol{x}) = \mathrm{KL}(\boldsymbol{x}, \boldsymbol{d})$ and proceeding as in Section 4.3.3, we see that in order to enforce the triangle inequality (i, k, j) we must update $\boldsymbol{x}$ as follows

$$\nabla\varphi(\boldsymbol{x}') = \nabla\varphi(\boldsymbol{x}) + \mu \boldsymbol{a}_{ikj} \quad \text{so that} \quad \langle \boldsymbol{a}_{ikj},\, \boldsymbol{x}' \rangle = 0. \tag{4.20}$$

It is easy to verify that $\nabla\varphi(\boldsymbol{x}) = \log \boldsymbol{x} - \log \boldsymbol{d}$. Since $\boldsymbol{a}_{ikj}$ is non-zero only in the positions corresponding to inequality (i, k, j), we can write (4.20) as

$$\begin{aligned} \log x'_{ik} &= \log x_{ik} + \mu \\ \log x'_{kj} &= \log x_{kj} - \mu \\ \log x'_{ji} &= \log x_{ji} - \mu, \\ \text{so that} \quad & x'_{ik} - x'_{kj} - x'_{ji} = 0. \end{aligned} \tag{4.21}$$

Thus, we obtain the solution

$$\mu = \frac{1}{2} \log \frac{x_{kj} + x_{ji}}{x_{ik}}. \tag{4.22}$$

From (4.22) we observe that $\mu > 0$ if the inequality (i, k, j) is satisfied strictly, $\mu = 0$ if there is equality, and $\mu < 0$ if the inequality is violated.

The corresponding algorithm for KL-Divergence metric nearness is given by Algorithm 4.4.

⋆4.4 Metric Nearness and APSP

NOTE: The material in this and the other starred sections below is provided for completeness. The material of this section is primarily the work of Justin Brickell, except for Lemma 4.4, which was originally suggested to us by [228] (though the proof given herein is our own).

The All Pairs Shortest Paths (APSP) problem [57] is an important and well-studied problem in graph theory that still continues to interest researchers. For a given weighted graph G, APSP aims to compute an associated matrix of distances M whose entry m_{ij} gives the weight of a shortest path between vertices i and j. A corresponding set of shortest paths between all vertices is also usually obtained.

On the surface, APSP appears to have no connection with the metric nearness problem. However, it turns out that APSP can be viewed as a special case of metric nearness. We develop this connection below. Note that in the previous sections we considered only symmetric matrices. However, in this section we consider asymmetric distance matrices, which are more natural for the APSP problem, as they correspond to directed graphs.

⋆4.4.1 Metric Nearness and APSP

Let the input be a weighted complete directed graph. We represent this graph by the (non-symmetric) matrix $\boldsymbol{D}$, where d_{ij} denotes the edge weight of edge (i, j). On $\boldsymbol{D}$ we perform a restricted version of metric nearness that permits only decreasing changes to the d_{ij} values. Curiously this decrease only

version of metric nearness is equivalent to APSP.

Lemma 4.4 formalizes this equivalence between a "decrease only" version of metric nearness and APSP. This equivalence was originally suggested by [228].

Lemma 4.4 (Decrease only metric nearness is APSP)**.** *Let $\boldsymbol{M}^A \in \mathscr{M}_N$ be the APSP solution for $\boldsymbol{D}$. Then, $\boldsymbol{M}^A$ is also the nearest "decrease only" metric solution. Furthermore, for any $\boldsymbol{M} \in \mathscr{M}_N$, if $\boldsymbol{M} \leq \boldsymbol{D}$ then $\boldsymbol{M} \leq \boldsymbol{M}^A$.*

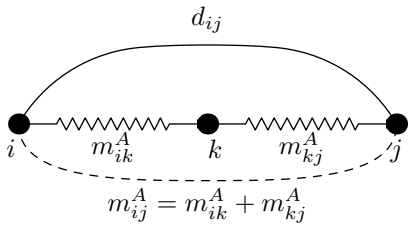


Figure 4.1: Shortest path between nodes i and j

Proof. We prove the last statement of the lemma, noting that it immediately implies the rest.

Assume the edge weights m_{ij}^A of $\boldsymbol{M}^A$ are sorted in increasing order, and that the least-weighted edge for which $\boldsymbol{M}$ exceeds $\boldsymbol{M}^A$ is m_{ij}, i.e., $m_{ij} > m_{ij}^A$. Since $\boldsymbol{M}^A$ is an APSP solution, each edge weight m_{ij}^A either equals d_{ij} or is the sum of weights of edges involved in a shortest path of length less than d_{ij}, as shown in Figure 4.1.

In the figure, k is some intermediate vertex on a shortest path from i to j. The zig-zag lines denote paths from $i \to k$ and $k \to j$. Since $\boldsymbol{M}$ is a metric

solution, $m_{ij} \leq m_{ik} + m_{kj}$. Now m^A_{ik}, $m^A_{kj} \leq m^A_{ij}$ because $m^A_{ij} = m^A_{ik} + m^A_{kj}$. By our assumption $m_{ij} > m^A_{ij}$ is the first place where a component of $\boldsymbol{M}$ exceeds a component of $\boldsymbol{M}^A$ (taken in sorted order), hence $m_{ik} \leq m^A_{ik}$ and $m_{kj} \leq m^A_{kj}$. Thus $m_{ij} \leq m^A_{ij}$. We have arrived at a contradiction to our initial assumption and that completes the proof of our claim. □

We take advantage of this relation to to provide a new linear-programming formulation for APSP, by solving a decrease only version of metric nearness (DOMN). For dense graphs APSP is commonly performed using the Floyd-Warshall algorithm, which has a complexity of $\Theta(n^3)$. Unlike the Floyd-Warshall algorithm that proceeds by fixing the triangles of the graph is a predetermined order, our DOMN algorithm fixes triangles in a data-dependent order. Empirically, our algorithm converges more quickly to the solution than Floyd-Warshall, despite having the same asymptotic worst-case behavior.

⋆4.4.2 The linear programming formulation of DOMN and its dual

Lemma 4.4 suggests that APSP solves the decrease only metric nearness problem regardless of the norm used to measure the error. We, however, focus on the ℓ_1 norm problem along with its linear programming formulation. The linear program is interesting both because it is a novel formulation for solving APSP, and also because its dual allows us to construct shortest paths, if desired. We apply the primal-dual technique for solving the resulting linear programs and obtain a new APSP algorithm as a consequence.

⋆4.4.2.1 Formulation

Let $\boldsymbol{X}$ represent a decrease-only distance matrix corresponding to the input matrix $\boldsymbol{D}$. Then the entries of $\boldsymbol{X}$ must satisfy,

$$x_{ij} \leq d_{ij} \qquad \text{for all } (i,j), \tag{4.23}$$

$$x_{ij} \leq x_{ik} + x_{kj} \quad \text{for all } (i,k,j). \tag{4.24}$$

Finding the matrix with the least ℓ_1 perturbation requires solving the problem

$$\underset{x_{ij}}{\text{Minimize}} \quad \sum_{ij}(d_{ij} - x_{ij}) \quad \text{subject to (4.23) and (4.24).}$$

Note that we are dealing directly with the values x_{ij} rather than the error values $e_{ij} = d_{ij} - x_{ij}$, as we did in sections 4.2.1 and 4.2.2. Since the d_{ij} are fixed we may replace this minimization by the equivalent problem

$$\begin{array}{lrcll} \underset{x_{ij}}{\text{Maximize}} & \sum_{ij} x_{ij} & & & \\ \text{subject to} & x_{ij} & \leq & d_{ij} & \text{for all } (i,j), \\ & x_{ij} - x_{ik} - x_{kj} & \leq & 0 & \text{for all } (i,k,j). \end{array} \tag{4.25}$$

The dual problem corresponding to (4.25) is

$$\begin{array}{lrcll} \underset{\delta_{ij}}{\text{Minimize}} & \sum_{ij} \delta_{ij} d_{ij} & & & \\ \text{subject to} & \delta_{ij} + \sum_{k \neq i,j} (\gamma_{ikj} - \gamma_{ijk} - \gamma_{kij}) & = & 1 & \text{for all } (i,j), \\ & \delta_{ij} & \geq & 0 & \text{for all } (i,j), \\ & \gamma_{ikj} & \geq & 0 & \text{for all } (i,k,j), \end{array} \tag{4.26}$$

where the dual variables δ_{ij} and γ_{ikj} correspond to the decrease-only constraints (4.23), and the triangle inequality constraints (4.24), respectively.

It is illustrative to cast the linear program (4.26) as a network flow problem, in which we must satisfy a demand for a single unit of flow between

every pair of vertices i and j. We can accomplish this either by sending the flow *directly* via the edge (i, j) (which corresponds to setting $\delta_{ij} = 1$) or by *bypassing* the edge (i, j) and routing through some other vertex k (which corresponds to setting $\gamma_{ikj} = 1$); in the latter case, we increase the demand for flow between (i, k) and (k, j) by 1.

We note that while there is a unique optimal solution to the linear program (4.25), the linear program (4.26) has several optimal solutions, some of which involve non-integral assignments to the γ_{ikj} variables. This non-uniqueness is not unexpected, because while there is only one shortest distance between two nodes in M, the shortest paths that achieve that distance may contain many intermediate nodes, each of which allows a γ_{ikj} variable to assume a positive assignment.

⋆4.4.3 A primal-dual algorithm for DOMN/APSP

We apply the primal-dual method [183, 217] to solve the linear programs for DOMN, and thereby obtain a new algorithm for APSP. Most treatments of the primal-dual method have a minimization of the primal problem and a maximization of the dual problem. Thus we will call (4.25) as the dual problem, and (4.26) as the primal problem. The primal-dual method begins with a feasible solution to the dual that is improved at each step by optimizing an *associated restricted primal* problem. In our case, we find it easier to optimize the associated restricted dual, whereby our method proceeds as follows:

1. Begin with a feasible solution to the dual problem. One such feasible solution is to set each x_{ij} to the smallest d_{ij} value.

2. Find the set P consisting of those constraints that do not have any additional slack. The decrease-only constraint $x_{ij} \leq d_{ij}$ (corresponding to dual variable δ_{ij}) will be in P iff $x_{ij} = d_{ij}$, and the triangle constraint $x_{ij} - x_{ik} - x_{kj} \leq 0$ (corresponding to dual variable γ_{ikj}) will be in P iff $x_{ij} = x_{ik} + x_{kj}$.

3. Find a solution to the associated restricted dual

$$\begin{array}{lrcll} \text{maximize} & \sum_{ij} u_{ij} & & & \\ \text{subject to} & u_{ij} & \leq & 0 & \text{if } \delta_{ij} \in P \\ & u_{ij} - u_{ik} - u_{kj} & \leq & 0 & \text{if } \gamma_{ikj} \in P \\ & u_{ij} & \leq & 1 & \text{for all } (i,j) \end{array} \tag{4.27}$$

4. If $u_{ij} = 0$ then the current value of x_{ij} is the optimal dual variable assignment; this result indicates that there are no variables that can be increased while still maintaining dual feasibility. Otherwise, improve the x_{ij} assignment by adding ϵu_{ij} to x_{ij}, where ϵ is as large as possible while still maintaining dual feasibility. Return to Step 2 with the new x_{ij} assignment.

By characterizing the solution of the associated restricted dual and the calculation of ϵ for the DOMN problem, we can give an efficient primal-dual algorithm. Observe that the solution to the associated restricted dual is that $u_{ij} = 1$ if the edge (i,j) is *increasable*, and 0 otherwise. Computing ϵ is equivalent to determining which of the increasable edges has the least capacity

ALGORITHM 4.5: Decrease-Only Metric Nearness: Simple $O(N^4)$ implementation

```
DOMN_ALG1(D)
Input: Dissimilarity matrix D
Output: M = APSP(D).
{Initialization}
u_ij ← d_ij for all i, j                          {Initial upper bounds}
x_ij ← min_{e'∈E} u_e'                            {Initial lower bounds}
I ← E                                             {Initial set of increasable edges}
while (I ≠ ∅)
   foreach (i, j) ∈ I with u_ij = x_ij
      I ← I − {(i, j)}                            {ij is no longer increasable}
      foreach k ≠ i, j
         u_ik = min(u_ik, u_ij + u_jk)            {Update upper bounds}
      end foreach
   end foreach
   foreach (i, j) ∈ I
      x_ij ← min_{e'∈I} u_e'                      {Update lower bounds}
   end foreach
end while
return M where m_ij = x_ij
```

for increase. Rather than use linear programming to determine the increasable edge set and computing ϵ explicitly, we can track upper bounds u_{ij} in addition to the lower bounds tracked by the x_{ij} variables. These upper bounds start as the d_{ij} values, but are reduced as edges become triangle constrained. Then the increasable set is simply the set of edges for which $x_{ij} < u_{ij}$, and ϵ is the difference between the lower bound of edges in the increasable set and the largest upper bound. Algorithm 4.5 implements these optimizations. I, the set of increasable edges, is the complement of P.

⋆4.4.4 Priority Queue DOMN Algorithm

Algorithm 4.5 requires $O(N^4)$ time, but we can do better by noticing that the only time the lower bounds are used is to check the condition $u_e = l_e$. For edges (i, j) not in I, we have $u_{ij} = l_{ij}$, whereas all edges (i, j) in I have l_{ij} equal to the smallest upper bound. Therefore, we can replace I with a priority queue ordered by upper bound, and we do not need to keep track of lower bounds at all (even though the original dual variable values x_{ij} were lower bounds). Algorithm 4.6 implements these changes, and requires only $O(N^3)$ time when implemented using a Fibonacci heap. Like the Floyd-Warshall algorithm, Algorithm 4.6 considers all edges in some order, and then fixes all triangles involving that edge. However, the Floyd-Warshall algorithm used a fixed data-independent order, whereas our algorithm uses a data-dependent order. As a result, our algorithm converges to the APSP/DOMN solution more rapidly, even though it still requires $O(N^3)$ time to complete.

⋆4.4.4.1 Equivalence of APSP to DOMN

Lemma 4.4 shows that the optimal assignment of the x_{ij} variables in linear program (4.25) is the same as the *distances* given by the APSP solution. In this section, we will investigate an equivalence between the optimal assignment of the δ_{ij} and γ_{ikj} variables in linear program (4.26) and the *paths* given by the APSP solution.

ALGORITHM 4.6: Decrease-Only Metric Nearness: Improved $O(N^3)$ Implementation

```
DOMN_ALG2(D)
Input: Dissimilarity matrix D
Output: M = APSP(D).
{Initialization}
u_ij ← d_ij  for all i, j                          {Initial upper bounds}
Q.ENQUEUE(ij, u_ij)  for all (i, j)                {Put all edges in priority-queue}
while (Q ≠ ∅)
   (i, j) ← Q.FIRST()                              {Remove edge with lowest upper bound}
   foreach k ≠ i, j
      u_ik = min(u_ik, u_ij + u_jk)                {Update upper bounds}
      Q.UPDATEPRIORITY(ik, u_ik)                   {Reorder priority queue}
   end foreach
end while
return M where m_ij = u_ij
```

DOMN from APSP Given an APSP solution, we construct an optimal solution to the DOMN problem (4.25) using the following procedure:

- If the edge (i, j) is used by n shortest paths, then set $\delta_{ij} = n$.
- If the edge (i, j) is used by n shortest paths *en route* to node k, then set $\gamma_{ijk} = n$.

Feasibility of the above assignment Clearly the non-negativity constraints $\delta_{ij} \geq 0$ and $\gamma_{ikj} \geq 0$ are satisfied. We must show that the constraint

$$\delta_{ij} + \sum_{k \neq i,j} (\gamma_{ikj} - \gamma_{ijk} - \gamma_{kij}) = 1$$

is satisfied for all edges (i, j).

For vertex pairs i and j in which the shortest path from i to j is the edge (i, j), this assignment will be consistent with the constraint because all shortest paths involving the edge (i, j) fall into one of three categories: the path from i to j, paths to j that end with the edge (i, j), and paths to another vertex k that pass through the edge (i, j). The latter two categories contribute both a $+1$ and a -1 to the constraint, while the first category contributes a $+1$, resulting in a net sum of 1.

For vertex pairs i and j in which the shortest path from i to j begins with the edge (i, k), this assignment is also consistent with the constraint. There are two types of shortest paths ending at node j and using edge (i, k): the path that starts at i, and paths that start at a node l and pass through (i, k) before finishing at j. The latter type of path contributes both a $+1$ and a -1 to the constraint, while the first type contributes a $+1$ for a total of 1.

Optimality of the assignment Under the proposed variable assignment procedure, the objective function for (4.26) is the sum of all path distances. Because the paths were taken from an APSP solution, this objective is minimized.

APSP from DOMN Given a optimal solution to (4.26), we construct an APSP solution using the following procedure:

- If δ_{ij} is positive, then the edge (i, j) is a shortest path from i to j.

- If γ_{ikj} is positive, then there is a shortest path from i to j that passes through k; we may recursively find the shortest paths from i to k and from k to j.

★4.5 An Application to Clustering

This application of Metric Nearness to clustering was primarily the work of Joel Tropp, and is included here for completeness.

The metric nearness problem can be used to develop efficient algorithms for clustering that provide guarantees on the quality of the output in comparison with the optimal clustering. The MAX-CUT problem offers an especially attractive example. A *cut* of a graph is a partition of the vertices into two disjoint sets, and the value of a cut is the total weight of all edges that cross the partition. MAX-CUT simply asks for the cut of a graph with maximum value. If the size of each edge weight is proportional to the dissimilarity between the two vertices, solving MAX-CUT can be interpreted as finding the best clustering of the vertices into two sets.

For a general set of weights, MAX-CUT is hard enough [218] that the solution cannot be well-approximated in polynomial time (unless $\mathsf{P} = \mathsf{NP}$) [3]. On the other hand, for weights that do satisfy the triangle inequality, de la Vega and Kenyon have exhibited a randomized algorithm that can approximate the solution arbitrarily well in polynomial time [67]. That is, for a given $\varepsilon > 0$, their method can (with high probability) compute in polynomial time, a cut whose value is no smaller than $(1 - \varepsilon)$ times the value of the optimal cut. Of

course, the time complexity grows quickly as ε shrinks.

Metric nearness plays an important role here. First, we approximate the original graph by a metric graph. Then, we use the fast algorithm to produce a nearly optimal cut of the metric graph. The same cut of the original graph also has a nearly optimal value, which can be bounded in terms of the approximation error from the metric nearness problem.

Theorem 4.5. *Suppose that $\boldsymbol{D}$ is a dissimilarity matrix and that $\boldsymbol{M}$ is a distance matrix. If $\mathcal{S}$ is a cut of $\boldsymbol{M}$ whose value exceeds $(1-\varepsilon)\,\mathrm{maxcut}(\boldsymbol{M})$, then we have the bounds*

$$\mathrm{cut}_{\mathcal{S}}(\boldsymbol{D}) \geq (1-\varepsilon)\,\mathrm{maxcut}(\boldsymbol{D}) - (1-\frac{\varepsilon}{2})\,\|\boldsymbol{M}-\boldsymbol{D}\|_1 \qquad \textit{and} \tag{4.28}$$

$$\mathrm{cut}_{\mathcal{S}}(\boldsymbol{D}) \geq \frac{1-\varepsilon}{\|\boldsymbol{M}/\boldsymbol{D}\|_\infty\,\|\boldsymbol{D}/\boldsymbol{M}\|_\infty}\,\mathrm{maxcut}(\boldsymbol{D}), \tag{4.29}$$

where '/' represents element-wise division and $\|\cdot\|_\infty$ denotes the ℓ_∞ norm that ignores the matrix diagonal. If $m_{jk} = d_{jk} = 0$, then the infinity norm also ignores the (j,k) entry of its argument.

To find the optimal $\boldsymbol{M}$ for bound (4.28), we simply solve the ℓ_1 metric nearness problem. The optimal $\boldsymbol{M}$ for (4.29) cannot be obtained without solving a non-convex optimization problem.

Proof. For a set of vertices $\mathcal{S}$, the value of the corresponding cut is computed by the linear function

$$\mathrm{cut}_{\mathcal{S}}(\boldsymbol{D}) = \sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} d_{jk}.$$

The maximum cut just optimizes this functional over all subsets $\mathcal{S}$ of the vertex set $\{1, 2, \ldots, n\}$:

$$\operatorname{maxcut}(\boldsymbol{D}) = \max_{\mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{k \notin \mathcal{S}} d_{jk}.$$

Obviously, $\operatorname{cut}_{\mathcal{S}}(\boldsymbol{D}) \leq \operatorname{maxcut}(\boldsymbol{D})$. It can be shown that $\operatorname{maxcut}(|\cdot|)$ is a matrix norm. In particular, it satisfies the triangle inequality for norms. It is also clear that

$$\operatorname{maxcut}(|\boldsymbol{T}|) \leq \tfrac{1}{2} \sum_{j \neq k} |t_{jk}| = \tfrac{1}{2} \|\boldsymbol{T}\|_1$$

for any symmetric matrix $\boldsymbol{T}$ with a zero diagonal.

Let us begin with bound (4.28). Suppose that $\mathcal{S}$ is a $(1 - \varepsilon)$-optimal cut of $\boldsymbol{M}$. Then

$$\begin{aligned}
\operatorname{cut}_{\mathcal{S}}(\boldsymbol{D}) &= \operatorname{cut}_{\mathcal{S}}(\boldsymbol{M}) + \operatorname{cut}_{\mathcal{S}}(\boldsymbol{D} - \boldsymbol{M}) \\
&\geq (1 - \varepsilon) \operatorname{maxcut}(\boldsymbol{M}) - \operatorname{cut}_{\mathcal{S}}(|\boldsymbol{D} - \boldsymbol{M}|) \\
&\geq (1 - \varepsilon) \operatorname{maxcut}(\boldsymbol{D} + (\boldsymbol{M} - \boldsymbol{D})) - \tfrac{1}{2} \|\boldsymbol{D} - \boldsymbol{M}\|_1 \\
&\geq (1 - \varepsilon) (\operatorname{maxcut}(\boldsymbol{D}) - \operatorname{maxcut}(|\boldsymbol{M} - \boldsymbol{D}|)) - \tfrac{1}{2} \|\boldsymbol{M} - \boldsymbol{D}\|_1 \\
&\geq (1 - \varepsilon) \operatorname{maxcut}(\boldsymbol{D}) - (1 - \varepsilon/2) \|\boldsymbol{M} - \boldsymbol{D}\|_1.
\end{aligned}$$

The proof for the bound (4.29) follows a similar outline. First, we implicitly define a relative error matrix $\boldsymbol{E}$ with the relation $\boldsymbol{M} = \boldsymbol{D} \odot \boldsymbol{E}$. We assume that $m_{jk} = 0$ if and only if $d_{jk} = 0$ to ensure $\boldsymbol{E}$ can be defined. If not, the resulting error bound would be trivial anyway. Let $r = \min\{e_{jk} : d_{jk} \neq 0\}$ and $R = \max\{e_{jk} : d_{jk} \neq 0\}$. For any zero entry of $\boldsymbol{D}$, take the corresponding

entry of $\boldsymbol{E}$ in the range $[r, R]$. In the sequel, we use '/' for element-wise division.

Next, observe that

$$\begin{aligned}\mathrm{cut}_{\mathcal{S}}(\boldsymbol{M}) = \mathrm{cut}_{\mathcal{S}}(\boldsymbol{D} \cdot \boldsymbol{E}) &= \sum_{j \in \mathcal{S}} \sum_{k \notin \mathcal{S}} d_{jk}\, e_{jk} \\ &\leq \max_{j \neq k} e_{jk} \sum_{j \in \mathcal{S}} \sum_{k \notin \mathcal{S}} d_{jk} \\ &\leq \|\boldsymbol{E}\|_\infty \, \mathrm{cut}_{\mathcal{S}}(\boldsymbol{D}).\end{aligned}$$

Similarly,

$$\mathrm{maxcut}(\boldsymbol{D}) = \mathrm{maxcut}(\boldsymbol{M}/\boldsymbol{E}) \leq \|\mathbf{1}/\boldsymbol{E}\|_\infty \, \mathrm{maxcut}(\boldsymbol{M}).$$

Then we compute

$$\begin{aligned}\mathrm{cut}_{\mathcal{S}}(\boldsymbol{D}) &\geq \frac{\mathrm{cut}_{\mathcal{S}}(\boldsymbol{M})}{\|\boldsymbol{E}\|_\infty} \\ &\geq \frac{1-\varepsilon}{\|\boldsymbol{E}\|_\infty} \, \mathrm{maxcut}(\boldsymbol{M}) \\ &\geq \frac{1-\varepsilon}{\|\boldsymbol{E}\|_\infty \, \|\mathbf{1}/\boldsymbol{E}\|_\infty} \, \mathrm{maxcut}(\boldsymbol{D}).\end{aligned}$$

This technique can be extended to other types of problems that are computationally easier for metric graphs [142]. Mettu and Plaxton have also considered fast algorithms for clustering "nearly metric" data, but their approach relies instead on weak versions of the triangle inequality [195]. Fast approximation algorithms for various other metric problems such as k-median, MAX-TSP, etc., are discussed in [143]; our method allows extending these approximation algorithms to non-metric data.

4.6 Experiments

We implemented metric nearness in C++ wherein we coded Algorithms 4.2 and 4.3. In this section we describe some experiments based on our implementation. All experiments were carried out on a P4/2.5GHz processor machine with 2GB RAM, running Linux.

4.6.1 Running Time Experiments

In this section we show some running time comparisons between CPLEX—a state of the art linear and quadratic optimization software—and our implementations of triangle fixing. Our results clearly indicate the overwhelming superiority of triangle fixing over CPLEX. For these experiments, we used random dissimilarity matrices of dimensions up to 100×100. The final values of the objective function achieved by CPLEX and our implementation agreed to five significant digits.

Figure 4.2 compares CPLEX quadratic programming to our implementation of ℓ_2 triangle fixing (see Algorithm 4.2). From the figure one can see that the triangle fixing procedure is up to thirty times faster than CPLEX's fastest method for solving the metric nearness quadratic program. Our experiments suggest that the ℓ_2 triangle fixing procedure scales as $O(N^3)$.

For ℓ_1 metric nearness, we compared CPLEX's fastest algorithm for metric nearness (determined by running all six choices available and selecting the fastest timing), and our implementation of the augmented triangle fixing procedure for solving the ℓ_1 metric nearness problem. Our implementation

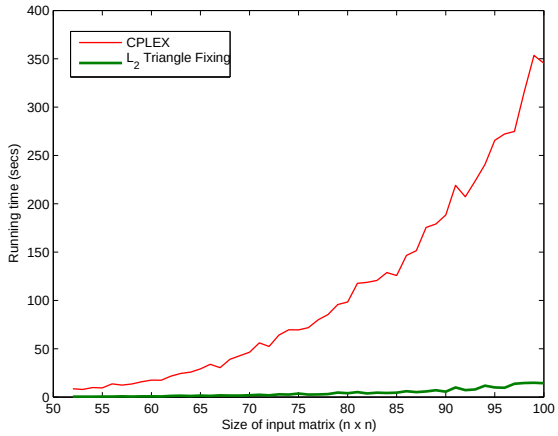


Figure 4.2: Running time comparison between CPLEX and the ℓ_2 triangle fixing algorithm.

runs up to 15 times faster than CPLEX, as indicated by Figure 4.3.

⋆4.6.2 Decrease only metric nearness/APSP experiments

Although Floyd-Warshall and the primal-dual Algorithm 4.6 both have an asymptotic runtime of $O(N^3)$, the latter more quickly converges to the answer for certain classes of problems. Floyd-Warshall chooses an order of triangles to correct without any guidance, whereas the primal-dual algorithm prefers to correct triangles that include shorter edges. We can certainly imagine a problem instance where the violating triangles have longer edges, and in this case the preference for shorter edges does not help.

For randomly generated test cases, however, our primal-dual algorithm did converge more quickly than Floyd-Warshall. To illustrate this observation, we generated random matrices of dimension 200×200 that had a zero diag-

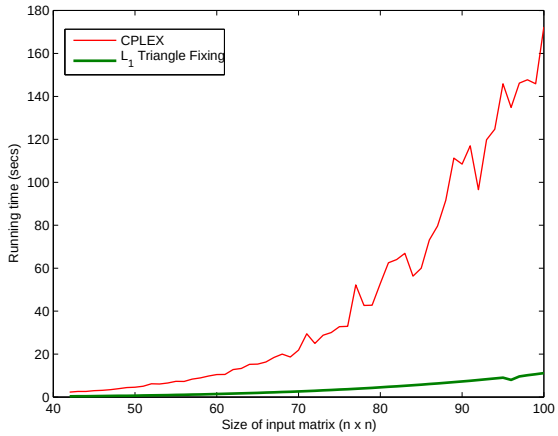


Figure 4.3: Running time comparison between CPLEX and augmented triangle fixing (ℓ_1).

onal and entries between 0.1 and 10. We then determined the correct answer before running both algorithms, halting the computation at each iteration to determine the distance between the current distance matrix and the final metric. Distance was computed as the l_1 vector distance. Figure 4.4 gives these results, which clearly show the primal-dual algorithm converging faster than Floyd-Warshall.

To determine the approximate time to convergence as a function of n, we generated $n \times n$ matrices for values of n from 25 to 225. Figure 4.5 plots the number of iterations required to converge for both Floyd-Warshall and the primal dual algorithm. The exponents in the big-O notation runtimes were approximated by fitting the curve to the best $a \cdot N^b$ approximation. While Floyd-Warshall takes the entire $O(N^3)$ time to converge, the primal-dual algorithm converges in about $O(N^{2.8})$ time. Even more striking is Figure 4.6,

which plots the number of iterations the algorithms required to *nearly* converge (where nearly converging means being within $0.5 * N$ of the metric solution). Here Floyd-Warshall still required $O(N^3)$ time, but the primal-dual algorithm needed only about $O(N^{2.5})$ time.

Unfortunately, we cannot yet take advantage of this rapid convergence to improve the runtime of the APSP primal-dual algorithms. Ideally, we could terminate the algorithm when we had modified enough edges to cause the graph to be a metric. After the graph is a metric, there are no triangles in violation, so the additional steps of the algorithm do not modify the graph in any way. However, we are unaware of any computationally efficient way to solve the problem of *metricity*. That is, given a graph, return "true" if the graph is a metric, and "false" otherwise. One way to solve this is to run APSP on the graph, and then check to see if any edges were shortened. This observation yields an upper bound of $O(APSP)$ on the metricity problem. It follows that we cannot terminate the APSP primal-dual algorithms early, even if they have converged to the correct result, because testing for the termination condition has the same complexity as the problem itself.

4.7 Discussion

Metric nearness is a rich problem. In this paper we formally introduced the problem and derived iterative algorithms for solving it for the vector ℓ_p norms. A special case of metric nearness was shown to be equivalent to the All Pairs Shortest Paths problem, which led to a new algorithm for APSP. We

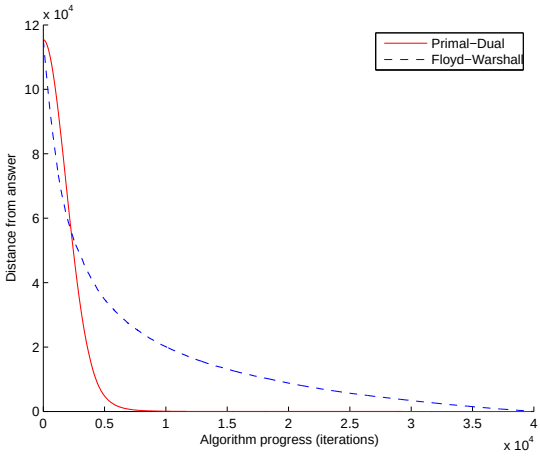


Figure 4.4: Convergence comparison of Floyd-Warshall and the primal-dual algorithm

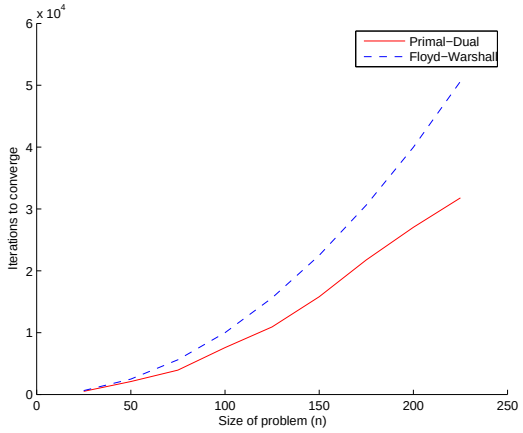


Figure 4.5: Iterations to converge for Floyd-Warshall and the primal-dual algorithm. Each iteration represents $O(N)$ computations.

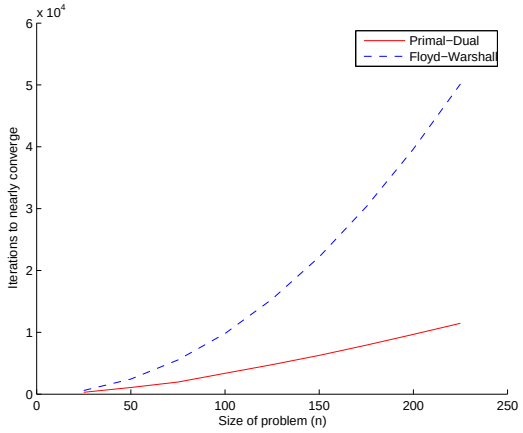


Figure 4.6: Iterations to nearly converge for Floyd-Warshall and the primal-dual algorithm. Each iteration represents $O(N)$ computations.

studied applications of metric nearness to MAX-CUT clustering. Experimental results illustrate the computational advantages of triangle fixing over generic optimization methods.

4.7.1 Variations

One may derive numerous variations of the metric nearness problem. The simplest of these involve the modification of the triangle inequality constraints in some interesting manners. These variations are all easily solved using our framework. Examples follow.

1. In Section 4.4 we discussed metric nearness with the restriction that permitted only decreasing changes to the entries of the input dissimilarity matrix. Similarly, one may also look at the problem where only increasing changes are permitted. Geometric or graph theoretic interpretations

of this problem remain to be considered.

2. When performing metric nearness on non-symmetric input graphs, one can choose either not to impose symmetry (as we did in the decrease-only section) or to impose symmetry. The latter case introduces additional constraints, but can be solved in our framework with only slight modifications.

3. Some applications may desire *rank* or order constraints to be enforced. That is, if the input satisfies $d_{ij} < d_{pq}$, then we also require $m_{ij} < m_{pq}$. Such a requirement can be useful in scenarios where the relative ordering of the dissimilarity values has a significance for the underlying application.

4. Box constraints, i.e., constraints of the type $l_{ij} \leq m_{ij} \leq u_{ij}$. Such constraints can be useful when a true metric, as opposed to a pseudo-metric, is desired (achieved by setting $l_{ij} > 0$). Upper bounds on the distance values may be utilized to prevent certain undesirable solutions.

5. Enforcement of λ-triangle inequalities that take the form $\lambda_{ij} m_{ij} \leq \lambda_{ik} m_{ik} + \lambda_{kj} m_{kj}$. Since the structure of the inequalities remains unaltered this problem can also be solved by triangle fixing.

Other variations involve generalization of the basic problem. The most important of such generalizations is one that introduces a weighting scheme

to the problem. Here we propose to obtain a distance matrix $\boldsymbol{M}$ such that

$$\boldsymbol{M} \in \operatorname*{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \|\boldsymbol{W} \odot (\boldsymbol{X} - \boldsymbol{D})\|,$$

where $\|\cdot\|$ is a norm, $\odot$ denotes the element-wise matrix product, and $\boldsymbol{W}$ is a weighting matrix (a symmetric nonnegative matrix). The weight matrix reflects our confidence in the entries of $\boldsymbol{D}$. When each d_{ij} represents a measurement with variance σ_{ij}^2, we might set $w_{ij} = 1/\sigma_{ij}^2$. If an entry of $\boldsymbol{D}$ is missing, one can set the corresponding weight to zero (however, the resulting problem loses strict convexity, whereby one should set this weight to a small value instead of zero).

4.7.2 Future work

Metric nearness is a relatively new problem. Many aspects could form a basis for future work and further consideration. The most immediate concerns that interest us are:

1. Extensions to triangle fixing; for example, one may speed up the procedure by fixing all the independent triangle inequalities in parallel. One could also attempt to fix a few dependent triangle inequalities at the same time, and such an approach will result in a dual block coordinate ascent scheme [277].

2. Studying the convergence of the triangle fixing algorithms at least for the ℓ_2 case. If possible, it would be interesting to furnish a proof of

convergence for our algorithms that is independent of the convergence of the more general Bregman's method.

3. Exploring applications of metric nearness, for e.g., applications to constrained clustering or various other applications that make use of proximity data and would profit from having metric data.

4.7.3 Open Problems

Two interesting open problems spring out of metric nearness. First is the metric verification problem that seeks to verify if the input dissimilarity matrix is actually a distance matrix. Some related work that probabilistically tests metric properties of an input dissimilarity matrix can be found in [220]. Whether the metric verification problem has the same complexity as the metric nearness problem remains to be ascertained. Second is the search for faster algorithms for the general metric nearness problem. Along with faster algorithms, the possibility of guaranteed polynomial time (non-iterative procedures) algorithms still remains.

4.7.4 Related work

Metric nearness is a relatively new problem that was introduced by the authors, where preliminary work includes [85, 86].

The most relevant research appears in recent papers of Roth et al. [240, 241]. They observe that machine learning applications often require metric data, and they propose a technique for converting general dissimilarity

data into metric data. Their method, constant-shift embedding, increases all the dissimilarities by an equal amount to produce a set of Euclidean distances (i.e., a set of numbers that can be realized as the pairwise distances among an ensemble of points in a Euclidean space). The size of the translation depends on the data, so the relative and absolute changes to the dissimilarity values can be large. Our approach is completely different. We seek a consistent set of distances that *deviates as little as possible* from the original measurements. In our approach, the resulting set of distances can arise from an arbitrary metric space; we do not restrict our attention to obtaining Euclidean distances. In consequence, we expect metric nearness to provide superior denoising. Moreover, our techniques can also learn distances that are missing entirely.

The technique of shifting the spectrum leads to an omission of the information carried by the negative eigenvalues of the input matrix. Laub and Müller [167] explore how the negative part of the spectrum could code for relevant features of the underlying data. Their method once again is based around computing an embedding, which is different from metric nearness, since the latter aims to only obtain a metric and constructs no embedding.

There are by now several papers discussing the problem of learning a parameterized Mahalanobis metric for underlying data, where the input is given in the form of pairwise similarity and dissimilarity values. For example the recent paper of Xing et al. [292], who learn a metric $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{G} (\boldsymbol{x}-\boldsymbol{y})}$, where $\boldsymbol{G}$ is an $s \times s$ positive semi-definite matrix. The user specifies that various pairs of points are similar or dissimilar. Then the

matrix $\boldsymbol{G}$ is computed by minimizing the total *squared* distances between similar points while forcing the total distances between dissimilar points to exceed one. The article provides explicit algorithms for the cases where $\boldsymbol{G}$ is diagonal and where $\boldsymbol{G}$ is an arbitrary positive semi-definite matrix. In comparison, the metric nearness problem is not restricted to Mahalanobis distances; it can learn a general discrete metric. It also allows us to use specific distance measurements and to indicate our confidence in those measurements (by means of a weight matrix), rather than forcing a binary choice of "similar" or "dissimilar." Some papers based on essentially the same framework for learning a Mahalanobis metric include [65, 105, 251, 256, 285].

The Metric Nearness Problem may appear similar to metric Multi-Dimensional Scaling [161], but we emphasize that the two problems are *distinct*. The latter problem endeavors to find an ensemble of points in a *prescribed* metric space—usually a Euclidean space—such that the distances between these points are close to the set of input distances. In contrast, metric nearness does not seek an embedding—it does not impose any hypotheses on the underlying space other than requiring it to be a metric space. For more details on Euclidean Distance Matrices see [108, 127, 250].

Related to metrics are ultrametrics; a distance matrix $\boldsymbol{M}$ is said to be an ultrametric if $m_{ij} \leq \max\{m_{ik}, m_{kj}\}$ for every distinct triple of indices (i, k, j). It is known that finding the nearest (in ℓ_1 and ℓ_2 norms) ultrametric to a given input matrix is NP-Complete [66]. However, the ℓ_∞-nearest ultrametric can be computed in $O(n^2)$ time [94]. Hubert *et al.* [135] consider the problem

of representing a dissimilarity matrix by a sum of matrices having a particular form, including a form that restricts the matrices to be ultrametrics.

Chapter 5

Decomposition Based Matrix Approximations

In Chapters 2 and 3 we derived iterative multiplicative update based methods for computing low-rank approximations, while in Chapter 4 we looked at a totally different matrix nearness problem. In this chapter we combine the ideas from these chapters to yield efficient algorithms for computing low-rank matrix approximations. Previously, we solved a class of low-rank approximations, namely, NNMA and incremental low-rank approximations, by constructing algorithms that at most guarantee a monotonic descent in the objective function value at each iteration. Such an approach yielded efficient and simple algorithms, though at the expense of some theoretical difficulties. If we solve the intermediate subproblems exactly, then we can make stronger guarantees on the convergence of the resultant algorithms and hopefully attain better local minima. Furthermore, we need not resort to heuristic procedures for dealing with regularization or additional convex constraints (as was done for some NNMA problems studied in Chapter 2).

The goal of this chapter is to provide efficient algorithms based on convex optimization for solving several low-rank matrix approximation problems. All the low-rank NNMA problems of the form (2.4) are subsumed as special

cases—though the algorithmic approach is totally different.

More formally, the low-rank approximation problem that we study in this chapter is

$$\min \quad \Delta(\boldsymbol{A},\, \boldsymbol{BC}), \quad \text{subject to } \boldsymbol{B} \in \mathcal{B}, \boldsymbol{C} \in \mathcal{C}. \tag{5.1}$$

We assume Δ to be a separable divergence measure, that is individually convex in $\boldsymbol{B}$ and $\boldsymbol{C}$. This requirement makes the set of problems somewhat smaller, and excludes most problems of the type (2.5) (except for the important Frobenius norm and KL-Divergence cases), however in return we get better theoretical guarantees and several new algorithms. The subproblems that arise as special cases of (5.1) are themselves of independent interest, for example, they include important problems such as Non-negative Least Squares (NNLS) and Maximum Entropy (MaxEnt).

5.1 Algorithmic Approach

We again choose the method of alternating minimization for obtaining a (locally optimal) solution to (5.1). Since the overall problem is non-convex, usually it is extremely hard to obtain a globally optimal solution (an exception is the SVD, where despite the non-convexity, one can easily compute a globally optimal solution). We use the following scheme (cf. page 26, Chapter 2).

ALGORITHM 5.7: Generic Alternating Descent Procedure

ALTERNATING_DESCENT($\boldsymbol{A}$, K)
Input: $\boldsymbol{A}$, K
Output: $\boldsymbol{B}$, $\boldsymbol{C}$
{Initialization}
Let $t \leftarrow 0$; Initialize $\boldsymbol{B}^0$ and/or $\boldsymbol{C}^0$
repeat
  $\boldsymbol{B}^{t+1} \leftarrow \text{argmin}_{\boldsymbol{B}}\ \Delta(\boldsymbol{A}, \boldsymbol{B}\boldsymbol{C}^t)$
  $\boldsymbol{C}^{t+1} \leftarrow \text{argmin}_{\boldsymbol{C}}\ \Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1}\boldsymbol{C})$
  $t \leftarrow t + 1$
until *convergence*.

The theorem below states that the alternating minimization procedure given above converges to a local-minimum of the objective function.

Theorem 5.1 (Convergence)**.** *If each intermediate step has a unique solution, then the sequence of iterates* $\{\boldsymbol{B}^t, \boldsymbol{C}^t\}$ *produced by the Algorithm 5.7 converges to a stationary point of the objective function* (5.1)*.*

Proof. Follows from the proof of convergence for a two-block Gauss-Seidel algorithm given by Grippo and Sciandrone [111]. □

5.1.1 Subproblems

Since we assumed Δ to be separable (in addition to being convex), we can restrict our attention to solving the following problem (with a slight abuse of notation)

$$\min_{\boldsymbol{c} \in \mathcal{C}} \quad \Delta(\boldsymbol{a}, \boldsymbol{B}\boldsymbol{c}) \tag{5.2}$$

for some arbitrary column $\boldsymbol{c}$ of $\boldsymbol{C}$ with the corresponding column $\boldsymbol{a}$ of $\boldsymbol{A}$. The matrix $\boldsymbol{B}$ is held fixed. Furthermore, we not only consider (5.2) but also the

important *regularized* variation of it, namely

$$\min_{\boldsymbol{c}\in\mathcal{C}} \quad \Delta(\boldsymbol{a},\, \boldsymbol{B}\boldsymbol{c}) + \beta(\boldsymbol{c}), \tag{5.3}$$

where $\beta(\boldsymbol{c}) \geq 0$ is some regularization function that can have several uses, for e.g., it can help to regulate $\boldsymbol{c}$ (make it more sparse, have bounded size, etc.), or it can help impart strict convexity to the minimization problem in case Δ is merely convex.

Note that since we assumed separability, $\Delta(\boldsymbol{A},\, \boldsymbol{B}\boldsymbol{C}) = \sum_n \Delta(\boldsymbol{a}_n,\, \boldsymbol{B}\boldsymbol{c}_n)$ and we can solve each of the N subproblems separately. However, since $\boldsymbol{B}$ is common to across these subproblems, we do not need to repeat computations dependent only on $\boldsymbol{B}$ across all subproblems. This point will become clearer when we describe a particular problem.

5.2 Solutions

In this section we present solutions to (5.2) and (5.3) for particular choices of Δ and β. We describe our generic technique by deriving solutions for the following three important special cases:

1. **Least-Squares:** Let $\Delta(\boldsymbol{a},\, \boldsymbol{B}\boldsymbol{c}) = \|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$. The main variations that we discuss are:

   (a) ordinary and regularized least-squares, i.e.,($\beta(\boldsymbol{c}) = \lambda\|\boldsymbol{c}\|_2^2$ and $\beta(\boldsymbol{c}) = \lambda\|\boldsymbol{c}\|_1$, $\lambda > 0$),

(b) nonnegative least squares ($\boldsymbol{c} \geq 0$), both ordinary and regularized, and

(c) ordinary and regularized least-squares problems with additional linear inequality constraints ($\boldsymbol{Gc} \leq \boldsymbol{h}$).

2. **KL-Divergence:** Let $\Delta(\boldsymbol{a}, \boldsymbol{Bc}) = \mathrm{KL}(\boldsymbol{a} \| \boldsymbol{Bc})$ or the asymmetric version $\mathrm{KL}(\boldsymbol{Bc} \| \boldsymbol{a})$. We discuss variations with $\beta(\boldsymbol{c}) = \|\boldsymbol{c}\|_1$ and $\beta(\boldsymbol{c}) = \|\boldsymbol{c}\|_2^2$, and special cases of MaxEnt where we minimize $\mathrm{KL}(\boldsymbol{Bc} \| \boldsymbol{1})$ subject to linear inequality constraints on $\boldsymbol{c}$.

3. **ℓ_1-norm:** Here $\Delta(\boldsymbol{a}, \boldsymbol{Bc}) = \|\boldsymbol{a} - \boldsymbol{Bc}\|_1$. We study both regularized and non-regularized versions of this problem. By itself, this subproblem arises in sparse approximation [275] and is a very important problem there (along with its least-distance version where one minimizes $\|\boldsymbol{c}\|_1$ subject to $\|\boldsymbol{a} - \boldsymbol{Bc}\| \leq \epsilon$. We look at all of these variations.

5.3 Least-squares

We begin by studying the regularized nonnegative least squares (NNLS) problem, remarking that the ordinary least squares problem is simpler and with a slight modification our method applies to it too. Recently Kim et al. [152] have developed a projected quasi-Newton method for solving the NNLS problem and they have applied it to the least-squares NNMA problem successfully. Here, we derive an NNLS method that is most beneficial for large to very large

scale problem. The optimization problem is

$$\min_{\boldsymbol{c}\geq 0} \quad \tfrac{1}{2}\|\boldsymbol{a}-\boldsymbol{B}\boldsymbol{c}\|_2^2+\beta(\boldsymbol{c}). \tag{5.4}$$

Clearly, (5.4) is a convex quadratic programming problem (when $\beta(\boldsymbol{c})$ is also convex), and it can be solved using a standard approach or off-the-shelf software. However, as the problem size becomes larger, most methods for solving (5.4) start becoming computationally too expensive and our aim here is to develop a scalable method for solving this problem.

5.3.1 NNLS ($\beta \equiv 0$)

First we consider the case without regularization. This is the standard nonnegative least-squares (NNLS) problem, which has been the subject of considerable interest in the literature [30, 40, 152, 171]. Surprisingly, despite being a fundamental and well-studied problem, NNLS does not seem to have good implementations for large-scale data, one exception being the recent work of Kim et al. [152]. Here we describe a method that scales to even larger problems that handled by [152] by making a potential sacrifice for accuracy by using a gradient based approach. The NNLS optimization problem is

$$\begin{aligned} \min \quad & f(\boldsymbol{c})=\tfrac{1}{2}\|\boldsymbol{a}-\boldsymbol{B}\boldsymbol{c}\|^2, \\ \text{subject to} \quad & \boldsymbol{c}\geq 0. \end{aligned} \tag{5.5}$$

At this point we introduce an auxiliary variable that plays a critical role in enabling an efficient solution to (5.5). Letting $\boldsymbol{y}=(\boldsymbol{B}\boldsymbol{c}-\boldsymbol{a})$, Problem (5.5)

can be rewritten as

$$\begin{aligned} \min \quad & f(\boldsymbol{y}, \boldsymbol{c}) = \tfrac{1}{2}\|\boldsymbol{y}\|^2 \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \qquad \boldsymbol{c} \geq 0. \end{aligned} \tag{5.6}$$

Now we form the Lagrangian of (5.6)

$$L(\boldsymbol{y}, \boldsymbol{c}, \boldsymbol{\nu}, \boldsymbol{\lambda}) = \tfrac{1}{2}\|\boldsymbol{y}\|^2 + \boldsymbol{\nu}^T(\boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y}) - \boldsymbol{\lambda}^T\boldsymbol{c},$$

where $\boldsymbol{\nu}$ and $\boldsymbol{\lambda}$ are dual variables. The dual function $g(\boldsymbol{\nu}, \boldsymbol{\lambda}) = \inf_{\boldsymbol{y},\boldsymbol{c}} L(\boldsymbol{y}, \boldsymbol{c}, \boldsymbol{\nu}, \boldsymbol{\lambda})$ is given by

$$g(\boldsymbol{\nu}, \boldsymbol{\lambda}) = -\boldsymbol{a}^T\boldsymbol{\nu} + \inf_{\boldsymbol{y}}(\tfrac{1}{2}\|\boldsymbol{y}\|^2 - \boldsymbol{\nu}^T\boldsymbol{y}) + \inf_{\boldsymbol{c}}(\boldsymbol{\nu}^T\boldsymbol{B}\boldsymbol{c} - \boldsymbol{\lambda}^T\boldsymbol{c}). \tag{5.7}$$

The second infimum in (5.7) is $-\infty$ unless $\boldsymbol{B}^T\boldsymbol{\nu} - \boldsymbol{\lambda} = 0$ (since it is a linear function of $\boldsymbol{c}$). The first infimum is achieved for $\boldsymbol{y} = \boldsymbol{\nu}$. Hence the resulting dual function is

$$g(\boldsymbol{\nu}, \boldsymbol{\lambda}) = \begin{cases} -\boldsymbol{a}^T\boldsymbol{\nu} - \tfrac{1}{2}\|\boldsymbol{\nu}\|^2, & \boldsymbol{B}^T\boldsymbol{\nu} - \boldsymbol{\lambda} = 0, \\ -\infty, & \text{otherwise.} \end{cases} \tag{5.8}$$

Thus the dual problem to (5.6) may be written as

$$\begin{aligned} \min \quad & g(\boldsymbol{\nu}, \boldsymbol{\lambda}) = \tfrac{1}{2}\|\boldsymbol{\nu}\|^2 + \boldsymbol{a}^T\boldsymbol{\nu} \\ \text{subject to} \quad & \boldsymbol{B}^T\boldsymbol{\nu} - \boldsymbol{\lambda} = 0, \quad \boldsymbol{\lambda} \geq 0, \end{aligned} \tag{5.9}$$

which can be replaced by the equivalent problem

$$\begin{aligned} \min \quad & g(\boldsymbol{\nu}) = \tfrac{1}{2}\|\boldsymbol{\nu}\|^2 + \boldsymbol{a}^T\boldsymbol{\nu} \\ \text{subject to} \quad & \boldsymbol{B}^T\boldsymbol{\nu} \geq 0. \end{aligned} \tag{5.10}$$

Problem (5.10) is now in a form that can be efficiently solved by a row-action procedure. We now leverage the ideas developed in §4.3.1 of Chapter 4 to

derive an algorithm for solving (5.10). Once we have obtained $\boldsymbol{\nu}^\star$, from the definition of the dual we know that $\boldsymbol{y}^\star = \boldsymbol{\nu}^\star = \boldsymbol{B}\boldsymbol{c}^\star - \boldsymbol{a}$. Hence, we now solve the the linear system $\boldsymbol{B}\boldsymbol{c}^\star = \boldsymbol{\nu}^\star + \boldsymbol{a}$ for $\boldsymbol{c}^\star$ using any large-scale iterative linear solver [196, 244]. Thus, our approach is highly scalable, because both (5.10) and the subsequent linear systems are solved via efficient large scale methods.

5.3.2 Regularized NNLS ($\beta(\boldsymbol{c}) = \frac{\lambda}{2}\|\boldsymbol{c}\|^2$)

Now we consider an ℓ_2-norm regularized version of NNLS. Here we have a slightly different approach, which exploits the regularization term to interleave the solution of the linear system with the row-action procedure itself. With the regularization function $\beta(\boldsymbol{c}) = \frac{\lambda}{2}\|\boldsymbol{c}\|^2$, the optimization problem (5.4) becomes

$$\begin{aligned} \min \quad & f(\boldsymbol{c}) = \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|^2 + \frac{\lambda}{2}\|\boldsymbol{c}\|^2, \\ \text{subject to} \quad & \boldsymbol{c} \geq 0. \end{aligned} \tag{5.11}$$

As before, introducing an auxiliary variable $\boldsymbol{y} = \boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}$, Problem (5.11) may be rewritten as

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\boldsymbol{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{c}\|^2, \\ \text{subject to} \quad & \boldsymbol{a} - \boldsymbol{B}\boldsymbol{c} - \boldsymbol{y} = 0, \qquad \boldsymbol{c} \geq 0. \end{aligned} \tag{5.12}$$

Subsequent to this change of variables (5.12) can be solved by a row-action procedure. We provide the pseudo-code in Algorithm 5.8.

ALGORITHM 5.8: Regularized NNLS (5.12).

RLS($\boldsymbol{B}$, $\boldsymbol{a}$, λ)
Input: $\boldsymbol{B}$, $\boldsymbol{a}$, λ
Output: Solution to (5.12)
{Initialization}
$\boldsymbol{c} \leftarrow \boldsymbol{0}$; $\boldsymbol{y} \leftarrow \boldsymbol{0}$; $\boldsymbol{z} \leftarrow \boldsymbol{0}$
while *not converged*
  {Enforce the hyperplane constraints}
  foreach $i \in [1..N]$
    $\theta_i \leftarrow (\|\boldsymbol{b}_i^T\|^2 + 1)^{-2}[\boldsymbol{b}_i^T\boldsymbol{c} - y_i - a_i]$ $(\star)$
    $y_i \leftarrow y_i + \theta_i$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} - \frac{\theta_i}{\mu}\boldsymbol{b}_i^T$
  end
  {Enforce non-negativity constraints} $(\star\star)$
  $\boldsymbol{t} \leftarrow \boldsymbol{c}$; $\boldsymbol{c} \leftarrow \max\{\boldsymbol{0}, \boldsymbol{c} - \boldsymbol{z}\}$; $\boldsymbol{z} \leftarrow \max\{\boldsymbol{0}, \boldsymbol{z} - \boldsymbol{t}\}$
end.
return $\{[\boldsymbol{y}; \boldsymbol{c}]\}$.

The step labeled $(\star)$ essentially computes the coefficient for orthogonal projection of $\boldsymbol{c}$ onto the i-th hyperplane $\boldsymbol{b_i^T c} - y_i = a_i$. Algorithm 5.8 can be easily extended to handle the case where $\boldsymbol{c}$ is subject to additional linear-inequality constraints, or simplified for solving just regularized least-squares (RLS) if needed. See the discussion below for more details.

5.3.2.1 Handling other constraints

By writing the subproblem in the form (5.6) we have gained two major benefits. The first benefit enables solution of large-scale problems, and the second is the ability to easily incorporate additional constraints. We remark that by following the row-action procedure of Dykstra [73], one can in fact,

handle arbitrary convex constraints. If the additional constraints are simple linear inequalities, then we can suitably modify Algorithm 5.8 itself. For example, say we have the constraints $\boldsymbol{G}\boldsymbol{c} \leq \boldsymbol{h}$, where G is $R \times K$, then for each $i \in [1..R]$ we perform the following steps in place of the line labeled ($\star\star$) in Algorithm 5.8.

$$\theta_i \leftarrow \|\boldsymbol{g}_i^T\|^{-2}[\boldsymbol{g}_i^T\boldsymbol{c} - h_i]^+; \quad \theta_i \leftarrow \min(z_i, \theta_i); \quad z_i \leftarrow z_i - \theta_i; \quad \boldsymbol{c} \leftarrow \boldsymbol{c} + \theta_i \boldsymbol{g}_i^T.$$

We remark that the projections described should be implemented to take advantage of the sparsity of the constraint matrices. However, this seems to be feasible only if we are solving the NNLS problem itself, rather than NNLS as a subproblem of NNMA (because in NNMA, the matrix $\boldsymbol{B}$ changes at each major iteration, and it is not easy to keep track of its sparsity pattern).

5.3.3 Sparse NNLS ($\beta(\boldsymbol{c}) = \lambda\|\boldsymbol{c}\|_1$)

The final interesting variant of NNLS that we study is the one most closely related to sparse approximations, and we call it sparse NNLS. Note that NNLS already produces sparse solutions due to the nonnegativity constraints on $\boldsymbol{c}$. However, further sparsity may be achieved by limiting the ℓ_1-norm of the optimization variable—a well established technique for favoring sparsity. The corresponding optimization problem is[1]

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2 + \lambda\|\boldsymbol{c}\|_1, \\ \text{subject to} \quad & \boldsymbol{c} \geq 0. \end{aligned} \tag{5.13}$$

[1]We remark that the methods of Chapter 2 yield a simple auxiliary function based method for solving (5.13). However, without further restrictions, it is not easy to guarantee convergence to the global minimum of (5.13) for those methods.

As for the other NNLS problems, we can introduce the auxiliary variable $\boldsymbol{y} = \boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}$ for this problem too. The resulting optimization problem is

$$\begin{aligned} \min \quad & f(\boldsymbol{y}, \boldsymbol{c}) = \tfrac{1}{2}\|\boldsymbol{y}\|^2 + \lambda\|\boldsymbol{c}\|_1, \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \qquad \boldsymbol{c} \geq 0. \end{aligned} \tag{5.14}$$

In problem (5.14), the presence of the ℓ_1-norm term raises a new problem, namely the inability to invoke a row-action procedure due to the non-strict convexity of the ℓ_1-norm (see related discussion in §4.3.2). We have studied such mixed ℓ_2-ℓ_1-norm optimization problems in the context of support vector machines (SVMs) elsewhere [264], and we can exploit those techniques to approximate the problem (5.14) by the following strictly convex optimization problem (where $\epsilon > 0$ is some user specified parameter)

$$\begin{aligned} \min \quad & f(\boldsymbol{y}, \boldsymbol{c}) = \frac{1}{2\epsilon}\|\boldsymbol{y}\|^2 + \lambda\|\boldsymbol{c} + \epsilon^{-1}\boldsymbol{1}\|_2^2, \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \qquad \boldsymbol{c} \geq 0. \end{aligned} \tag{5.15}$$

The problem (5.15) can be now easily solved via a row-action procedure. We omit the details for brevity.

We could also follow the approach via the dual of (5.14), as we did in §5.3.1. In this case, it can be seen after some algebra that the dual problem corresponding to (5.14) is

$$\begin{aligned} \min_{\boldsymbol{\nu}} \quad & \tfrac{1}{2}\|\boldsymbol{\nu}\|^2 + \boldsymbol{\nu}^T\boldsymbol{a}, \\ \text{subject to } & \boldsymbol{B}^T\boldsymbol{\nu} \geq -\lambda\boldsymbol{1}. \end{aligned} \tag{5.16}$$

Problem (5.16) can be solved easily by a row-action procedure and subsequently we can recover the optimum $\boldsymbol{c}$ from $\boldsymbol{\nu}$, since $\boldsymbol{y} = \boldsymbol{a} - \boldsymbol{B}\boldsymbol{c} = \boldsymbol{\nu}$, as

before. Problem (5.16) might be more preferable because it does not require the selection of an extra regularization parameter ϵ. However, it does require the subsequent solution of a linear system, which might make it unsuitable from an implementation perspective. Both versions, however, are highly scalable.

5.4 KL-Divergence

This unregularized problem cannot be solved in the primal by introducing an auxiliary variable $\boldsymbol{y} = \boldsymbol{Bc}$ because just as for (5.14) it lacks strict convexity, a necessary ingredient for the application of a row-action procedure. The primal problem is

$$\min_{\boldsymbol{c} \geq 0} \quad \mathrm{KL}(\boldsymbol{a} \| \boldsymbol{Bc}). \tag{5.17}$$

However, due to its special structure, we again pass over to the dual as shown below. The Lagrangian for (5.17) after letting $\boldsymbol{y} = \boldsymbol{Bc}$ is

$$L(\boldsymbol{y}, \boldsymbol{c}, \boldsymbol{\nu}, \boldsymbol{\lambda}) = \mathrm{KL}(\boldsymbol{a} \| \boldsymbol{y}) + \boldsymbol{\nu}^T(\boldsymbol{Bc} - \boldsymbol{y}) - \boldsymbol{\lambda}^T \boldsymbol{c}. \tag{5.18}$$

From (5.18) we obtain the corresponding dual function

$$g(\boldsymbol{\nu}, \boldsymbol{\lambda}) = \begin{cases} \inf_{\boldsymbol{y}} \mathrm{KL}(\boldsymbol{a} \| \boldsymbol{y}) - \boldsymbol{\nu}^T \boldsymbol{y} & \text{if } \boldsymbol{B}^T \boldsymbol{\nu} = \boldsymbol{\lambda}, \\ -\infty & \text{otherwise.} \end{cases} \tag{5.19}$$

The infinum in (5.19) is attained for $\boldsymbol{y} = \boldsymbol{a}/(\boldsymbol{1} - \boldsymbol{\nu})$. Thus, the dual problem is

$$\begin{aligned} \max_{\boldsymbol{\nu}, \boldsymbol{\lambda}} \quad & \sum_i a_i \log(1 - \nu_i), \\ \text{subject to} \quad & \boldsymbol{B}^T \boldsymbol{\nu} = \boldsymbol{\lambda} \geq 0. \end{aligned}$$

Replacing $\boldsymbol{\nu} \leftarrow \mathbf{1} - \boldsymbol{\nu}$, and simplifying the constraint we obtain the equivalent problem

$$\begin{aligned} \max_{\boldsymbol{\nu}} \quad & \sum_i a_i \log \nu_i, \\ \text{subject to} \quad & \boldsymbol{B}^T \boldsymbol{\nu} \leq \boldsymbol{B}^T \mathbf{1}. \end{aligned} \tag{5.20}$$

Algorithm 5.9 solves (5.20). Let $\boldsymbol{\nu}^*$ denote the optimal solution of (5.20), then we can obtain the optimal solution $\boldsymbol{c}$ to the original problem by solving $\boldsymbol{B}\boldsymbol{c} = \boldsymbol{a}/\boldsymbol{\nu}^*$.

Algorithm 5.9: KL-Divergence Minimization (5.20).

KLMin($\boldsymbol{B}$, $\boldsymbol{a}$)
Input: $\boldsymbol{B}$, $\boldsymbol{a}$
Output: Solution to (5.20)
{Initialization}
$\boldsymbol{\nu} \leftarrow \mathbf{1}$; $\boldsymbol{z} \leftarrow \mathbf{0}$
while *not converged*
  {Enforce the hyperplane constraints}
  foreach $i \in [1..N]$
    $\theta_i \leftarrow (\|\boldsymbol{b}_i^T\|^2 + 1)^{-2}[\boldsymbol{b}_i^T \boldsymbol{c} - y_i - a_i]$ $(\star)$
    $y_i \leftarrow y_i + \theta_i$
    $\boldsymbol{c} \leftarrow \boldsymbol{c} - \frac{\theta_i}{\mu} \boldsymbol{b}_i^T$
  end
  {Enforce non-negativity constraints} $(\star\star)$
  $\boldsymbol{t} \leftarrow \boldsymbol{c}$; $\boldsymbol{c} \leftarrow \max\{\mathbf{0}, \boldsymbol{c} - \boldsymbol{z}\}$; $\boldsymbol{z} \leftarrow \max\{\mathbf{0}, \boldsymbol{z} - \boldsymbol{t}\}$
end.
return $\{[\boldsymbol{y}; \boldsymbol{c}]\}$.

An approach via the dual however is not so simple for the asymmetric problem

$$\min_{\boldsymbol{c} \geq 0} \quad \mathrm{KL}(\boldsymbol{B}\boldsymbol{c} \| \boldsymbol{a}), \tag{5.21}$$

because one needs to solve difficult nonlinear equations at each individual projection step. However, a direct primal approach with the auxiliary variable $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}$ is not feasible either because of the subsequent lack of strict convexity in the objective function. However, in the presence of a strictly convex regularization term such as $\lambda\|\boldsymbol{c}\|_2^2$, the problem (5.21) becomes amenable to our techniques. We skip the details for brevity.

5.5 The ℓ_1-norm

To avoid repetition, we describe our approach for the following regularized ℓ_1-norm based optimization problem. The unregularized case by be solved by going over to the dual as shown in the previous sections. The optimization problem is

$$\begin{aligned} \min_{\boldsymbol{c},\boldsymbol{y}} \quad & g(\boldsymbol{y},\boldsymbol{c}) = \|\boldsymbol{y}\|_1 + \mu\|\boldsymbol{c}\|_1, \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \quad \text{and} \quad \boldsymbol{c} \geq 0. \end{aligned} \tag{5.22}$$

The problem (5.22) again lacks strict convexity and we again invoke the techniques successfully applied in Section 4.3.2 to overcome this problem. Introducing a variable $\boldsymbol{z} = |\boldsymbol{y}|$ (5.22) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{y},\boldsymbol{c}} \quad & \boldsymbol{1}^T\boldsymbol{z} + \boldsymbol{0}^T\boldsymbol{y} + \mu\boldsymbol{1}^T\boldsymbol{c}, \\ \text{subject to} \quad & \boldsymbol{z} = |\boldsymbol{y}|, \quad \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \quad \text{and} \quad \boldsymbol{c} \geq 0. \end{aligned}$$

Introducing the conformal vectors $\boldsymbol{w} = [\boldsymbol{z};\boldsymbol{y};\boldsymbol{c}]$ and $\boldsymbol{d} = [\boldsymbol{1};\boldsymbol{0};\mu\boldsymbol{1}]$, we can finally invoke Mangasarian's [1984] theorem and arrive at the quadratic program

$$\begin{aligned} \min_{\boldsymbol{w}=[\boldsymbol{z};\boldsymbol{y};\boldsymbol{c}]} \quad & \|\boldsymbol{w} + \epsilon\boldsymbol{d}\|^2, \\ \text{subject to} \quad & \boldsymbol{z} = |\boldsymbol{y}|, \quad \boldsymbol{B}\boldsymbol{c} - \boldsymbol{a} - \boldsymbol{y} = \boldsymbol{0}, \quad \text{and} \quad \boldsymbol{c} \geq 0, \end{aligned} \tag{5.23}$$

where $\epsilon > 0$ is some constant that relates (5.22) to (5.23). This quadratic program can be now easily solved via an appropriate modification to Algorithm 5.8.

5.6 The general case

The subproblem for the general case is (again we restrict our attention to the regularized case for brevity)

$$\begin{aligned} \underset{\boldsymbol{y},\boldsymbol{c}}{\text{minimize}} \quad & g(\boldsymbol{y},\boldsymbol{a}) = \Delta(\boldsymbol{y},\boldsymbol{a}) + \mu\beta(\boldsymbol{c}), \\ \text{subject to} \quad & \boldsymbol{B}\boldsymbol{c} - \boldsymbol{y} = 0, \quad \text{and } \boldsymbol{c} \geq 0. \end{aligned} \tag{5.24}$$

Then we solve the following equations

$$\begin{aligned} \nabla_{\boldsymbol{y}} g(\boldsymbol{y}^{k+1}, \boldsymbol{c}^k) &= \nabla_{\boldsymbol{y}} g(\boldsymbol{y}^k, \boldsymbol{c}^k) - \theta_i \boldsymbol{e}^i, \quad \text{for each } i, \\ \nabla_{\boldsymbol{c}} g(\boldsymbol{y}^{k+1}, \boldsymbol{c}^{k+1}) &= \nabla_{\boldsymbol{c}} g(\boldsymbol{y}^k, \boldsymbol{c}^k) + \theta_i \boldsymbol{b}^i, \quad \text{for each } i. \end{aligned}$$

The vector $\boldsymbol{e}^i$ denotes the i-th standard basis vector and $\boldsymbol{b}^i$ denotes the i-th row of matrix $\boldsymbol{B}$.

Chapter 6

EM via Matrix Nearness with Application to Clustering Directional Data

We introduced the notion of parametric mixture modeling in §1.2.2, wherein we showed how traditional mixture modeling may be viewed as a matrix nearness problem. In this chapter, we pick up that thread again and show the well-known auxiliary function based derivation of the Expectation Maximization (EM) algorithm; subsequently we apply EM to the problem of clustering directional data. At this point, we would like to draw the reader's attention to the fact that even though the auxiliary function view/genesis of EM is not new, our derivation is amongst the simplest presentations of EM.

6.1 EM via Matrix Nearness

In this section we rewrite the problem of parametric mixture modeling as a matrix nearness problem—the main goal of which is to highlight the auxiliary function technique of Chapter 2 to naturally arrive at a simple derivation of the EM algorithm. This connection indicates that our matrix approximation problems of Chapter 2 can benefit from the theory and algorithms that have been developed in association with EM over the years, and conversely

some of our techniques can be carried over for obtaining new EM procedures.

Let the input be the set of vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$ (also called the observed or incomplete data in EM parlance) that are assumed to have been sampled from some underlying probability density. The goal of traditional mixture modeling is to parametrize a set of possible densities, and given the observed data samples, to estimate the associated set of parameters. Under appropriate assumptions on the parameter estimation process (e.g., maximum likelihood or Bayesian estimation) standard results from statistics indicate that in the presence of a sufficiently large number of input samples, the estimated model parameters approach the true underlying parameters.

In traditional *maximum likelihood* mixture modeling our aim is to learn the model (expected incomplete data density)

$$p(\boldsymbol{a}|\boldsymbol{\Theta}) = \sum_{k=1}^{K} p(\boldsymbol{a}|\boldsymbol{\theta}_k, k)P(k), \tag{6.1}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$ is the set of parameters that characterize the model. The contribution of the k-th component is given by $P(k)$ ($\sum_k P(k) = 1$), and for simplicity we assume that the $\boldsymbol{\theta}_k$ are independent. Thus, given the set of (i.i.d.) input vectors $[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ we estimate $\boldsymbol{\Theta}$ by maximizing the likelihood

$$p(\boldsymbol{A}|\boldsymbol{\Theta}) = \prod_{n=1}^{N} p(\boldsymbol{a}_i|\boldsymbol{\Theta}), \tag{6.2}$$

with respect to $\boldsymbol{\Theta}$. It is more convenient to maximize the *log-likelihood*

$$L(\boldsymbol{A}; \boldsymbol{\Theta}) = \sum_{n} \log p(\boldsymbol{a}_i|\boldsymbol{\Theta}), \tag{6.3}$$

instead of (6.3). Using (6.1) we can rewrite (6.3) as

$$L(\boldsymbol{A};\boldsymbol{\Theta}) = \sum_{n=1}^{N} \log\Big(\sum_{k=1}^{K} P(k)p(\boldsymbol{a}_n|\boldsymbol{\theta}_k,k)\Big). \tag{6.4}$$

Writing $p_{nk} = p(\boldsymbol{a}_n|\boldsymbol{\theta}_k)$ and $\alpha_k = P(k)$, the problem of maximizing (6.4) may be rewritten as the matrix nearness problem

$$\min_{\boldsymbol{\Theta},\boldsymbol{\alpha}} \quad \mathrm{KL}(\mathbf{1}\|\boldsymbol{P}\boldsymbol{\alpha}) - \big(\sum_{n}(\boldsymbol{P}\boldsymbol{\alpha})_n - 1\big), \tag{6.5}$$

where $\sum_k \alpha_k = 1$, and $\boldsymbol{\Theta}$ is subject to appropriate constraints depending on what functional form is assumed for $p(\boldsymbol{a}|\boldsymbol{\theta}_k)$. From (6.5) it now becomes immediate that maximizing the log-likelihood (6.4) is essentially equivalent to computing a regularized maximum entropy solution for the parameters.

Maximizing (6.4) (or minimizing (6.5)) directly can be quite difficult. Hence, as before, we can simplify the problem by constructing an auxiliary function for $\mathcal{L}(\boldsymbol{A};\boldsymbol{\Theta})$ that is easier to maximize. Recall that if $Q(\boldsymbol{\Theta};\widehat{\boldsymbol{\Theta}})$ is an auxiliary function for $\mathcal{L}(\boldsymbol{A};\boldsymbol{\Theta})$ then

1. $\mathcal{L}(\boldsymbol{A};\boldsymbol{\Theta}) = Q(\boldsymbol{\Theta};\boldsymbol{\Theta})$ for all $\boldsymbol{\Theta}$, and
2. $\mathcal{L}(\boldsymbol{A};\boldsymbol{\Theta}) \geq Q(\boldsymbol{\Theta};\widehat{\boldsymbol{\Theta}})$ for all $\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}$.

The trick is to of course construct an auxiliary function Q that is easier to optimize than $\mathcal{L}$. Once we have Q, then we immediately have an iterative procedure for maximizing $\mathcal{L}$, and that is exactly what the well-known Expectation Maximization (EM) algorithm does. Monotonic ascent in the log-likelihood is

assured by maintaining

$$\mathcal{L}(\boldsymbol{A}; \boldsymbol{\Theta}^{t+1}) = Q(\boldsymbol{\Theta}^{t+1}; \boldsymbol{\Theta}^{t+1}) \geq Q(\boldsymbol{\Theta}^{t+1}; \boldsymbol{\Theta}^{t}) \geq Q(\boldsymbol{\Theta}^{t}; \boldsymbol{\Theta}^{t}) = \mathcal{L}(\boldsymbol{\Theta}^{t}),$$

where the first inequality follows from Property 2 of auxiliary functions, and the second inequality is due to the update

$$\boldsymbol{\Theta}^{t+1} = \operatorname*{argmax}_{\boldsymbol{\Theta}} \; Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{t}).$$

Below we show how to easily construct an auxiliary function Q for $\mathcal{L}$. We remark at this point that Csiszar and Tusnady [61] viewed EM as an alternating minimization procedure, where the E-step essentially minimizes a KL-Divergence to obtain an estimate for the distribution, and the M-step estimates the optimum parameters holding the distribution fixed. In auxiliary function terminology, the E-step computes the auxiliary function $Q(\cdot; \boldsymbol{\Theta}^{t})$, and the M-step subsequently optimizes it. We describe these details below.

6.1.1 The E-step

Exploiting the concavity of $\log x$, the log-likelihood (6.4) may be bounded as

$$\mathcal{L}(\boldsymbol{A}; \boldsymbol{\Theta}) = \sum_{i} \log(\boldsymbol{P}\boldsymbol{\alpha})_i \geq \sum_{ik} \beta_{ik} \log \frac{\alpha_k p_{ik}}{\beta_{ik}}, \tag{6.6}$$

where $\beta_{ik} \geq 0$, and $\sum_k \beta_{ik} = 1$. In other words (6.6) yields the auxiliary function

$$Q(\boldsymbol{\Theta}; \widehat{\boldsymbol{\Theta}}) = \sum_{ik} \beta_{ik} \log \frac{\alpha_k p(\boldsymbol{a}_i | \boldsymbol{\theta}_k, k)}{\beta_{ik}}, \tag{6.7}$$

where β_{ik} is given by

$$\beta_{ik} = \frac{\hat{\alpha}_k p(\boldsymbol{a}_i|k, \hat{\boldsymbol{\theta}}_l)}{\sum_l \hat{\alpha}_l p(\boldsymbol{a}_i|l, \hat{\boldsymbol{\theta}}_l)}. \tag{6.8}$$

It is easy to verify that (6.7) is an auxiliary function for $\mathcal{L}$. In the language of the EM algorithm, $\beta_{ik} = P(k|\boldsymbol{a}_i, \boldsymbol{\theta}_k)$, i.e., the posterior probability for component k within the mixture. Later we will talk about a different choice of β_{ik}, namely the *hard-assignment* heuristic, wherein for a given data point $\boldsymbol{a}_i$, the coefficient β_{ik} is zero for all k except one, essentially indicating that data point $\boldsymbol{a}_i$ "belongs" to component k in the mixture. Note that by viewing the β_{ik} values as arbitrary convex coefficients, the E-step can be easily generalized. The resulting procedure will then fall under the Generalized Alternating Minimization view of EM taken by Gunawardana and Byrne [117].

Note that the above derivation of the auxiliary function is independent of the actual functional form assumed by the densities $p(\boldsymbol{a}|\boldsymbol{\theta})$. Naturally, the maximization procedure depends on p, and forms the so-called M-step of the EM algorithm. Depending on the actual form of $p(\boldsymbol{a}|\boldsymbol{\theta})$ the M-steps vary significantly across different probability distributions, and sometimes pose significant parameter estimation challenges.

We will now leave our general discussion on EM, and proceed to model the input data as directional or unit ℓ_2-norm data. We will derive efficient parameter updates, for the so-called M-step of EM for two particular distributions, noting that these derivations are non-trivial and do not just follow directly from a canned application of the M-step. We provide below some

background information about directional data and distributions, before the actual derivations.

6.2 Directional Data

There are several domains where methods based on minimizing Euclidean distortions yield poor results [268]. For example, studies in information retrieval applications convincingly demonstrate *cosine similarity* to be a more effective measure of similarity for analyzing and clustering text documents using a discriminative approach such as single-link or complete-link hierarchical clustering. In this domain, there is substantial empirical evidence that normalizing the data vectors helps to remove the biases induced by the length of a document and provide superior results [246, 247]. Further, the spherical kmeans (`spkmeans`) algorithm [77], that performs kmeans using cosine similarity instead of Euclidean distortion, has been found to work well for text clustering. Datasets from such domains, where similarity measures such as cosine, Jaccard or Dice [235] are more effective than measures derived from Mahalanobis type distances, possess intrinsic "directional" characteristics, and are hence better modeled as *directional data* [187].

There are many other important domains such as bioinformatics (see for e.g., Eisen et al. [93]), collaborative filtering (for e.g., Sarwar et al. [248]) etc., in which directional data is encountered. Consider the Pearson correlation coefficient, which is a popular similarity measure in both these domains . Given $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, the Pearson product moment correlation between them is given by

$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{d}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{d}(x_i-\bar{x})^2}\times\sqrt{\sum_{i=1}^{d}(y_i-\bar{y})^2}}$, where $\bar{x} = \frac{1}{d}\sum_{i=1}^{d} x_i$, $\bar{y} = \frac{1}{d}\sum_{i=1}^{d} y_i$. Consider the mapping $\boldsymbol{x} \mapsto \tilde{\boldsymbol{x}}$ such that $\tilde{x}_i = \frac{x_i-\bar{x}}{\sqrt{\sum_{i=1}^{d}(x_i-\bar{x})^2}}$, and a similar mapping for $\boldsymbol{y}$. Then we have $\rho(\boldsymbol{x}, \boldsymbol{y}) = \tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{y}}$. Moreover, $\|\tilde{\boldsymbol{x}}\|_2 = \|\tilde{\boldsymbol{y}}\|_2 = 1$. Thus, the Pearson correlation is exactly the cosine similarity between $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$. Hence, analysis and clustering of data using Pearson correlations is essentially a clustering problem for directional data.

Below we discuss generative mixture-models based on the von Mises-Fisher (vMF) and the Watson distributions for modeling directional data. The vMF distribution is the "natural" distribution for directional data [187], while the Watson distribution provides a natural model for axially symmetric directional data.

For both the vMF and Watson distributions we derive efficient EM algorithms; the E-step was derived in §6.1.1 above, and the main contribution comes from our derivation of the M-step. For both the vMF and Watson distributions this step turns out to be quite challenging, requiring the solution of difficult nonlinear equations. We derive efficient numerical approximations for solving these nonlinear equations and point out that this estimation is non-trivial, and is crucial for an efficient implementation.

As an outcome of our EM algorithms for doing mixture modeling with both these densities, we formulate clustering procedures, which in themselves are very interesting, because they yield a theoretical basis for both spherical kmeans [77] (via vMF) and diametric kmeans [84] (via Watson). These connections provide deeper insight into the success of both of these variants

of kmeans, and also lend theoretical validity to the popular cosine similarity measure.

We apply our resultant algorithms to the problem of clustering high-dimensional text and gene-expression data based on mixtures of vMF and Watson distributions, obtaining results superior to common approaches for clustering these data.

We would like to point out that at the time of writing this dissertation, other work performing mixture modeling for Watson distributions has appeared in the literature [29]. However, the author's of [29] were not aware of the relation of mixtures of Watson distributions to diametric clustering, which we describe in this paper. The parameter estimates derived by them are also different from our approach (in fact [29] follow our approach for vMF densities to obtain their parameter estimates).

6.2.1 Directional Distributions: Background

In this section we summarize some background material about directional distributions to provide an introduction to the uninitiated reader. Those who are familiar with these basics can straightway skip to the next section. The material in this section is based upon [187], though all the proofs are of our own construction.

Let $\mathbb{S}^{M-1}$ denote the M-dimensional unit hypersphere, i.e., $\mathbb{S}^{M-1} = \{\boldsymbol{a}|\boldsymbol{a} \in \mathbb{R}^M, \text{and } \|\boldsymbol{a}\|_2 = 1\}$. All the densities that we describe will be defined on the surface of this unit hypersphere. We denote the probability

element on $\mathbb{S}^{M-1}$ by $d\mathbb{S}^{M-1}$, and parameterize $\mathbb{S}^{M-1}$ by going over to polar coordinates $(r, \boldsymbol{\theta})$, where $r = 1$, and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{M-1}]$. Consequently $a_m = \sin\theta_1 \cdots \sin\theta_{m-1} \cos\theta_m$ for $1 \leq m < M$, and $x_M = \sin\theta_1 \cdots \sin\theta_{M-1}$. Given this parameterization, it is easy to show that $d\mathbb{S}^{M-1} = \left(\prod_{k=2}^{d-1} \sin^{d-k}\theta_{k-1}\right) d\boldsymbol{\theta}$ (see §B.1).

6.2.2 Uniform distribution

The uniform distribution on $\mathbb{S}^{M-1}$ has its probability element equal to $c_M d\mathbb{S}^{M-1}$, where c_M is the normalization constant such that

$$\int_{\mathbb{S}^{M-1}} c_M d\mathbb{S}^{M-1} = 1.$$

Performing this simple integration (see §B.4.1), we obtain

$$c_M = \Gamma(M/2)/2\pi^{M/2}.$$

6.2.3 The von Mises-Fisher distribution

A unit norm random vector $\boldsymbol{a}$ is said to have the M-dimensional von Mises-Fisher (vMF) distribution if its probability element is $c_M(\kappa) e^{\kappa \boldsymbol{\mu}^T \boldsymbol{a}} d\mathbb{S}^{M-1}$, where $\|\boldsymbol{\mu}\| = 1$ and $\kappa \geq 0$. The normalizing constant

$$c_M(\kappa) = \frac{\kappa^{M/2-1}}{(2\pi)^{M/2} I_{M/2-1}(\kappa)},$$

where $I_s(\kappa)$ denotes the modified Bessel function of the first kind (see §B.4.2 for a derivation). Note that traditionally, researchers in directional statistics normalize the integration measure by the uniform measure, so that instead

of $c_M(\kappa)$, one uses $c_M(\kappa)2\pi^{M/2}/\Gamma(M/2)$—as far as parameter estimation is concerned, this distinction is immaterial, and we shall ignore it for the rest of this chapter.

The vMF density $p(\boldsymbol{a}|\boldsymbol{\mu},\kappa) = c_M(\kappa)e^{\kappa\boldsymbol{\mu}^T\boldsymbol{a}}$ is parameterized by the mean direction $\boldsymbol{\mu}$, and the *concentration* parameter κ, so-called because it characterizes how strongly the unit vectors drawn according to $p(\boldsymbol{a}|\boldsymbol{\mu},\kappa)$ are concentrated about the mean direction $\boldsymbol{\mu}$. Larger values of κ imply stronger concentration about the mean direction. In particular when $\kappa = 0$, $p(\boldsymbol{a}|\boldsymbol{\mu},\kappa)$ reduces to the uniform density on $\mathbb{S}^{M-1}$, and as $\kappa \to \infty$, $p(\boldsymbol{a}|\boldsymbol{\mu},\kappa)$ tends to a point density.

The vMF distribution is one of the simplest general distributions for directional data, and has properties analogous to those of the multi-variate Gaussian distribution for multi-variate data in $\mathbb{R}^M$. For example, the maximum entropy density on $\mathbb{S}^{M-1}$ subject to the constraint that $E[\boldsymbol{a}]$ be fixed, is a vMF density (see Rao [233], pp. 172–174 and Mardia [186] for details).

6.2.4 Watson distribution

The uniform and the vMF distributions are defined over *directions*. However, sometimes the observations are *axes*, wherein the vectors $\boldsymbol{a}$ and $-\boldsymbol{a}$ are indistinguishable [187]. To model such axial data one of the simplest densities is the Watson density, whose probability element is given by $c_M(\kappa)e^{\kappa(\boldsymbol{\mu}^T\boldsymbol{a})^2}d\mathbb{S}^{M-1}$. After integrating (see §B.4.3) to determine the constant

we find

$$c_M(\kappa) = \frac{\Gamma(M/2)}{2\pi^{M/2}\,{}_1F_1(\frac{1}{2}, \frac{d}{2}, \kappa)}, \tag{6.9}$$

where ${}_1F_1$ denotes a confluent Hypergeometric function, also known as Kummer's function (see [1]). Due to the $e^{\kappa(\boldsymbol{\mu}^T\boldsymbol{a})^2}$ term in the Watson density, for $\kappa > 0$, the distribution tends to concentrate around $\pm\boldsymbol{\mu}$ as κ increases, whereas for $\kappa < 0$, the density concentrates around the great circle orthogonal to $\boldsymbol{\mu}$. Since $(\boldsymbol{Q}\boldsymbol{\mu}^T\boldsymbol{Q}\boldsymbol{x})^2 = (\boldsymbol{\mu}^T\boldsymbol{x})^2$ for any orthogonal matrix $\boldsymbol{Q}$, the Watson density is rotationally invariant.

6.3 Parameter estimation or the M-step

Now we are ready to show the details of the parameter estimation or M-step of EM as applied to mixtures of vMF and Watson distributions. In this step we maximize (6.6) w.r.t. the parameters $\boldsymbol{\theta}_k$, while keeping β_{ik} fixed. This amounts to solving

$$\max_{\boldsymbol{\alpha},\boldsymbol{\Theta}} \quad \sum_{ik} \beta_{ik} \log \alpha_k p(\boldsymbol{a}_i|\theta_k), \tag{6.10}$$

subject to $\boldsymbol{\Theta} \in \Omega$, where Ω is the space of parameters. We assume that for $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$ the individual per component parameters $\boldsymbol{\theta}_k$ are independent of each other. Hence, the maximization (6.10) is essentially a concatenation of K different maximization problems.

Recall that α_k is the prior for the k-th class; we maximize (6.10) w.r.t. α_k subject to the restriction that $\sum_k \alpha_k = 1$ to obtain

$$\alpha_k = \frac{1}{N} \sum_i \beta_{ik}. \tag{6.11}$$

The main challenge is posed by the estimation of $\boldsymbol{\Theta}$. We derive below such estimates for both the vMF and Watson densities.

6.3.1 M-step for Mixture of vMFs

For the vMF distribution the parameters to be estimated are $\kappa_k \geq 0$, and $\boldsymbol{\mu}_k$ for $1 \leq k \leq K$. Maximizing (6.10) w.r.t. $\boldsymbol{\mu}_k$ subject to the restriction $\boldsymbol{\mu}_k^T \boldsymbol{\mu}_k = 1$, we obtain

$$\boldsymbol{\mu}_k = \frac{\sum_i \beta_{ik} \boldsymbol{a}_i}{\| \sum_i \beta_{ik} \boldsymbol{a}_i \|}. \tag{6.12}$$

Similarly, differentiating (6.10) w.r.t. κ_k and solving we obtain

$$\kappa_k \quad = \quad \text{Solve} \left[\frac{I_{M/2}(\kappa_k)}{I_{M/2-1}(\kappa_k)} \quad = \quad \frac{\| \sum_i \beta_{ik} \boldsymbol{a}_i \|}{\sum_i \beta_{ik}} \quad = \quad \bar{r} \right], \tag{6.13}$$

where we use $\bar{r}$ to essentially denote the average *resultant* vector length (cf. (6.12)).

Solving for κ_k requires the solution of the nonlinear equation (6.13), as indicated by the Solve[·] operation. However, for high-dimensional data, a nonlinear root-finder for solving (6.13) can be very time consuming and somewhat of an engineering challenge to implement due to various numerical issues (e.g. problems such as overflow due to the potentially huge magnitude of $I_{M/2}$). In Section 6.4.1 we bypass these problems by obtaining a closed-form approximation for κ_k that solves (6.13).

6.3.2 M-step for Mixture of Watson Distributions

For the Watson distribution the parameters to be estimated are $\boldsymbol{\mu}_k$ and κ_k for $1 \leq k \leq K$. Maximizing (6.10) w.r.t. $\boldsymbol{\mu}_k$ subject to the restriction

$\boldsymbol{\mu}_k^T \boldsymbol{\mu}_k = 1$, we obtain

$$\boldsymbol{\mu}_k = \quad \frac{\sum_i \beta_{ik} \boldsymbol{a}_i \boldsymbol{a}_i^T \boldsymbol{\mu}_k}{\| \sum_i \beta_{ik} \boldsymbol{a}_i \boldsymbol{a}_i^T \boldsymbol{\mu}_k \|} = \frac{\boldsymbol{K}_k \boldsymbol{\mu}_k}{\| \boldsymbol{K}_k \boldsymbol{\mu}_k \|}, \tag{6.14}$$

where $\boldsymbol{K}_k = \sum_i \beta_{ik} \boldsymbol{a}_i \boldsymbol{a}_i^T$. Similarly, differentiating (6.10) w.r.t. κ_k and solving we obtain

$$\kappa_k = \quad \text{Solve} \left[\frac{{}_1F_1'(\frac{1}{2}, \frac{M}{2}, \kappa_k)}{{}_1F_1(\frac{1}{2}, \frac{M}{2}, \kappa_k)} \quad = \quad \frac{\sum_i \beta_{ik} (\boldsymbol{a}_i^T \boldsymbol{\mu}_k)^2}{\sum_i \beta_{ik}} \quad = \bar{r} \right], \tag{6.15}$$

where $\bar{r}$ denotes the average resultant vector length (cf. (6.14)). Observe that in (6.14) the variable $\boldsymbol{\mu}_k$ occurs on both sides of the equation, thereby necessitating an iterative solution, which is easily seen to be given by the leading left-singular vector of the matrix $\boldsymbol{K}_k$. Computing κ_k requires the solution of the difficult nonlinear equation (6.15), as indicated by the Solve[$\cdot$] operation. As for the vMF case, solving (6.15) using a nonlinear root-finder is impractical because of numerical difficulties and engineering issues. In Section 6.4.3 we derive numerical approximations for computing κ_k which are not only extremely efficient, but also avoid making even a single function computation!

6.4 Approximating κ

Recall that because of the lack of an analytical solution, it is not possible to directly solve for the κ_k values, and one must either use a nonlinear root-finder or obtain some asymptotic approximation for solving (6.13) and (6.15). However, high-dimensional data poses additional challenges because it is not easy to even compute Bessel or Hypergeometric functions without

resorting to multi-precision floating point libraries. Even then, a root finder is iterative and several expensive iterations are required for obtaining the solution. Therefore, a good numerical approximation to κ_k is highly desirable, and we derive such approximations for both the vMF and Watson distributions here.

6.4.1 Estimating κ for vMFs

Mardia and Jupp [187] provided approximations for estimating κ for a given value of $\bar{r}$, for the following two limiting cases (Approximations 10.3.7 and 10.3.10 of Mardia and Jupp [187], pp. 198):

$$\hat{\kappa} \approx \frac{M-1}{2(1-\bar{r})} \qquad \text{valid for large } \bar{r}, \quad (6.16)$$

$$\hat{\kappa} \approx d\bar{r}\left(1 + \frac{M}{M+2}\bar{r}^2 + \frac{M^2(d+8)}{(M+2)^2(M+4)}\bar{r}^4\right) \qquad \text{valid for small } \bar{r}, \quad (6.17)$$

where $\bar{r}$ is as defined in (6.13).

Both these approximations assume that $\kappa \gg M$, which is typically not valid for high dimensions. Also, the $\bar{r}$ values corresponding to several real-world text and gene expression datasets usually lie in the mid-range rather than towards the extreme ranges catered to by the above approximations. We obtain a more accurate approximation for κ as described below.

Let $A_M(\kappa)$ denote the ratio $I_{M/2}/I_{M/2-1}$ that occurs in (6.13). The important observation is that $A_M(\kappa)$ is a ratio of Bessel functions that differ

in their order by one. For such a ratio there exists the following continued fraction representation [284]. Letting $s = M/2 - 1$ we have

$$A_{2s+2}(\kappa) = \frac{I_{s+1}(\kappa)}{I_s(\kappa)} = \cfrac{1}{\frac{2(s+1)}{\kappa} + \cfrac{1}{\frac{2(s+2)}{\kappa} + \cdots}}. \tag{6.18}$$

Letting $A_{2s+2}(\kappa) = \bar{r}$ we can write (6.18) approximately as

$$\frac{1}{\bar{r}} \approx \frac{2(s+1)}{\kappa} + \bar{r},$$

which gives the approximation,

$$\kappa \approx \frac{(2s+2)\bar{r}}{1-\bar{r}^2}.$$

This approximation to κ was not found to be empirically that accurate, and we empirically determined a correction term of $-\bar{r}^3/(1-\bar{r}^2)$, adding which leads to the approximation

$$\hat{\kappa} = \frac{\bar{r}M - \bar{r}^3}{1-\bar{r}^2}. \tag{6.19}$$

This approximation could be made even more accurate by adding other correction terms that are functions of $\bar{r}$ and M [1]. For other approximations of κ (including the derivations of (6.16) and (6.17)) and some related issues please see §B.2.3.

[1] Note that if one wants a more accurate approximation, it is easier to use (6.19) as a starting point and then perform Newton-Raphson iterations for solving $A_M(\hat{\kappa}) - \bar{r} = 0$, since it is easy to evaluate $A'_M(\kappa) = 1 - A_M(\kappa)^2 - \frac{M-1}{\kappa} A_M(\kappa)$. However, for high-dimensional data, this method is impractical because computing the ratio $A_M(\kappa)$ itself is difficult and requires extended precision floating point arithmetic as the Bessel functions grow rapidly.

6.4.2 Experimental study of the approximation

In this section we provide a brief experimental study to assess the quality of our approximation of the concentration parameter κ. We previously mentioned that for large values of $\bar{r}$ ($\bar{r}$ close to 1), approximation (6.16) is reasonable; for small values of $\bar{r}$ (usually for $\bar{r} < 0.2$) estimate (6.17) is quite good; Eqn. (6.19) yields good approximations for most values of $\bar{r}$.

To properly assess the quality of our approximation and compare it with (6.16) and (6.17), first note that a particular value of $\bar{r}$ may correspond to many different combinations of κ and d values. We present experimental results to evaluate the accuracy of the approximations over the parts of the M-κ plane that are expected to be encountered in the target application domains. However, we begin by comparing the accuracies at a scattering of points on this plane via Table 6.1 which shows the actual numerical values of κ that the three approximations (6.16), (6.17), and (6.19) yielded at these points. The $\bar{r}$ values shown in the table were computed using (6.13) for a single vMF.

$(M, \bar{r}, \kappa)$	$\hat{\kappa}$ = Eq. (6.16)	$\hat{\kappa}$ = Eq. (6.17)	$\hat{\kappa}$ = Eq. (6.19)
$(10, 0.633668, 10)$	12.2839	9.36921	**10.1631**
$(100, 0.46945, 60)$	93.2999	59.3643	**60.0833**
$(500, 0.46859, 300)$	469.506	296.832	**300.084**
$(1000, 0.554386, 800)$	1120.92	776.799	**800.13**

Table 6.1: Approximations $\hat{\kappa}$ for a sampling of κ and M values.

To supplement Table 6.1, which showed how the three approximations behave on a sampling of points from the (κ, M) plane, in this section we present

experimental results some slices of this plane, where we either keep M fixed and vary κ, or we keep κ fixed and vary M.

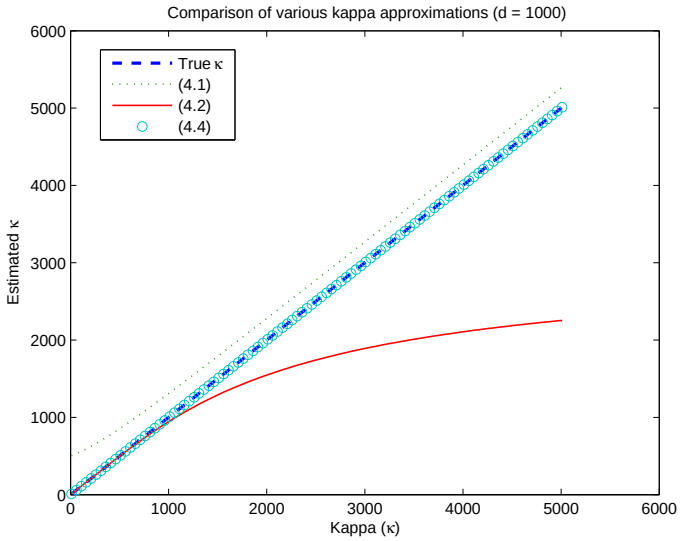


Figure 6.1: Comparison of true and approximated κ values ($M = 1000$).

We begin by holding M fixed at 1000, and allow κ to vary from 10 to 5010. Figure 6.1 shows the values of computed $\hat{\kappa}$) (estimation of κ) using the three approximations. From this figure one can see that (6.16) overestimates the true κ, while (6.17) underestimates it. However, our approximation (6.19) is very close to the true κ values.

Next we illustrate the quality of approximation when κ is held fixed and M is allowed to vary. Figure 6.2 illustrates how the various approximations behave as the dimensionality M is varied from $M = 4$ to $M = 1454$. The concentration parameter κ was set at 500 for this experiment. We see that (6.17) catches up with the true value of κ after approximately $M \geq 2\kappa$

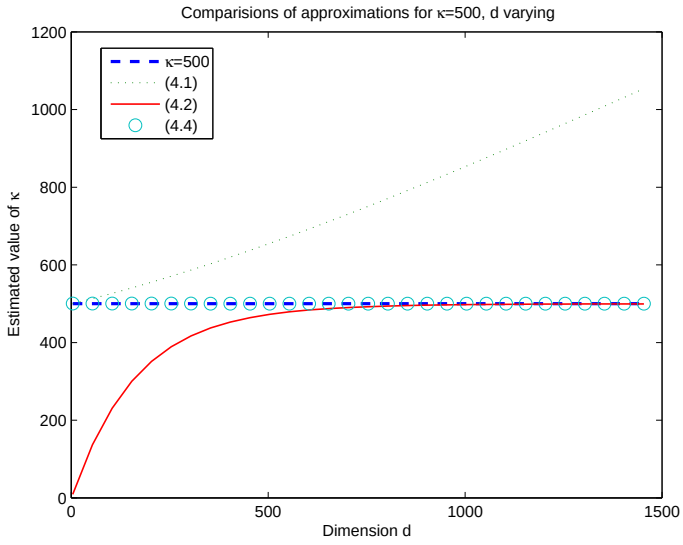


Figure 6.2: Comparison of approximations for varying M ($\kappa = 500$).

(because the associated $\bar{r}$ values become small), whereas (6.19) remains accurate throughout.

Since all the approximations depend on $\bar{r}$ (which implicitly depends on κ and M), it is illustrative to also plot the approximation errors as $\bar{r}$ is allowed to vary. Figure 6.3 shows how the three approximations perform as $\bar{r}$ ranges from 0.05 to 0.95. Let $f(M, \bar{r})$, $g(M, \bar{r})$, and $h(M, \bar{r})$ represent the approximations to κ using (6.16), (6.17) and (6.19), respectively. Figure 6.3 displays $|A_M(f(M, \bar{r})) - \bar{r}|$, $|A_M(g(M, \bar{r})) - \bar{r}|$, and $|A_M(h(M, \bar{r})) - \bar{r}|$ for the varying $\bar{r}$ values. Note that the y-axis is on a log-scale to appreciate the differences between the three approximations. We see that up to $\bar{r} \approx 0.18$ (dashed line on the plot), the approximation yielded by (6.17) has lower error. Thereafter, approximation (6.19) becomes better.

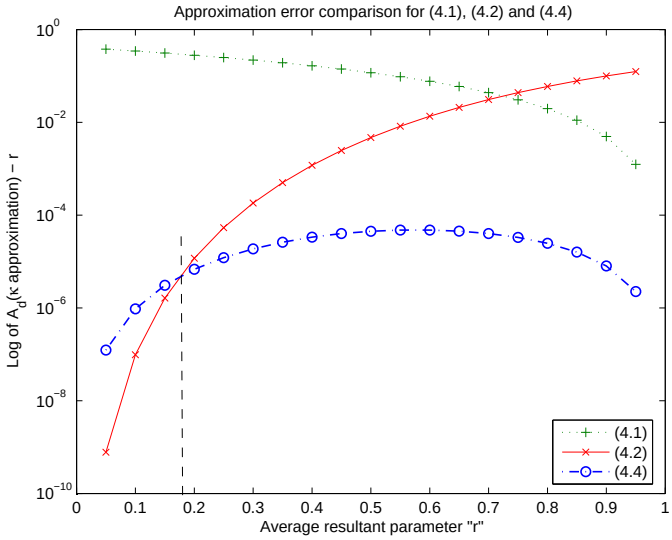


Figure 6.3: Comparison of approximations for varying $\bar{r}$ ($M = 1000$).

6.4.3 Approximating κ for Watson

In this section we exploit some of the properties of the confluent hypergeometric function ${}_1F_1$ to obtain an extremely efficient approximation to (6.15). We remark that Bijral et al. [29] have obtained good approximations for κ by following our continued fraction expansion approach of §6.4.2. We provide a some news approximation in this section. It is well known [1] that

$$\frac{\partial\, {}_1F_1(a,b,z)}{\partial z} = \frac{a}{b}\, {}_1F_1(a+1,b+1,z). \tag{6.20}$$

Assuming that b is relatively large we approximate (6.20) to write

$$\frac{a}{b}\, {}_1F_1(a+1,b+1,z) \approx \frac{a}{b-1}\, {}_1F_1(a+1,b,z), \tag{6.21}$$

essentially replacing b by $b-1$. A useful identity that ${}_1F_1$ satisfies is

$$(a+z)\,{}_1F_1(a+1,b,z)+(b-a-1)\,{}_1F_1(a,b,z)+(1-b)\,{}_1F_1(a+1,b-1,z)=0. \tag{6.22}$$

We approximate the last term in the identity above by ${}_1F_1(a+1,b,z)$. Hence we get the new approximation

$$(a+b-1+z)\,{}_1F_1(a+1,b,z)\approx(a+b-1)\,{}_1F_1(a,b,z). \tag{6.23}$$

Recall that in (6.15) we needed to essentially solve

$$\frac{{}_1F_1'(a,b,z)}{{}_1F_1 a,b,z}=\bar{r},$$

where a, b, z, and $\bar{r}$ are defined appropriately. Using (6.21) in (6.23) we obtain

$$(a+b-1+z)\frac{b-1}{a}\,{}_1F_1'(a,b,z)\approx(a+b-1)\,{}_1F_1(a,b,z). \tag{6.24}$$

We solve this latter approximation (writing ${}_1F_1'(a,b,z)/\,{}_1F_1(a,b,z)=\bar{r}$) to obtain

$$z\approx\frac{a(a+b-1)}{(b-1)\bar{r}}. \tag{6.25}$$

However, in practice we have observed that the "corrected"-approximation

$$z\approx(a+b-1)\left(\frac{1}{1-\bar{r}}-\frac{a}{(b-1)\bar{r}}\right), \tag{6.26}$$

leads to much better accuracy. This accuracy may be viewed as the result of incorporating the relative error term

$$\epsilon=\frac{{}_1F_1'(a,b,z)}{{}_1F_1'(a,b,z)-\,{}_1F_1(a,b,z)},$$

into (6.24), so that we solve the "corrected"-approximation

$$((a+b-1)(1+\epsilon)+z)\frac{b-1}{a}\,{}_1F_1'(a,b,z) \approx (a+b-1)\,{}_1F_1(a,b,z). \qquad (6.27)$$

This solution of (6.27) is given by (6.26), and it yields significantly better accuracy than (6.27) in practice.

6.4.4 A more careful look at the approximations

It is obvious that the error of approximation depends heavily upon the parameters a, b and z. Depending upon these parameters, we have the following four approximations (all of these are variations of (6.26)).

$$z \approx (a+b-1)\frac{1}{1-\bar{r}} \qquad (A1)$$

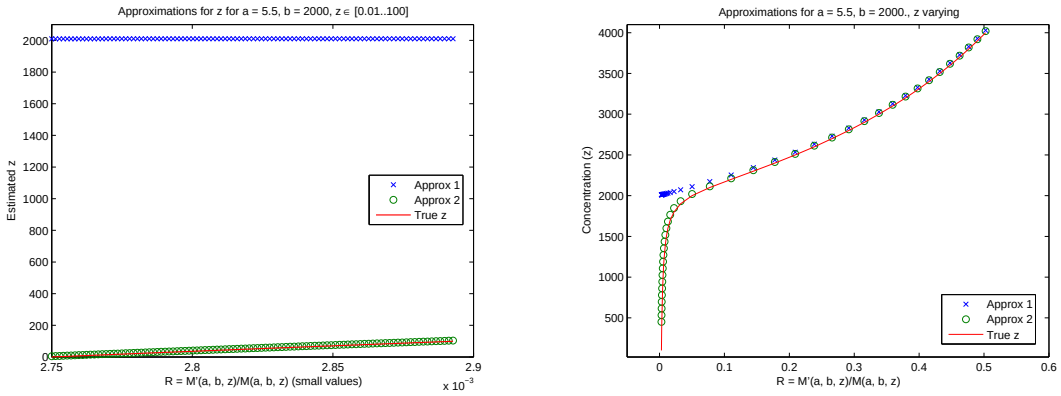
$$z \approx (a+b-1)\left(\frac{1}{1-\bar{r}} - \frac{a}{(b-1)\bar{r}}\right) \qquad (A2)$$

$$z \approx (a+b-1)\left(\frac{1}{1-\bar{r}} + \frac{a-1}{(b-1)\bar{r}}\right) \qquad (A3)$$

$$z \approx (a+b-1)\left(\frac{1}{1-\bar{r}} - \frac{a}{b\bar{r}}\right). \qquad (A4)$$

We conducted some experiments to determine the parameter ranges for which these approximations work well. Figure 6.4 displays the behavior of the approximations (A1) and (A2) across a range of z as a and b are held fixed. We display $\bar{r} = {}_1F_1'/\,{}_1F_1$ on the X-axis, since the approximations are functions of $\bar{r}$, and show varying degrees of accuracy for small or large values of $\bar{r}$.

Figure 6.5 reports results similar to those in Figure 6.4 except that now $a = 0.5$ is used, whereby to attain better approximations we had to use (A3). In this case (A2) leads to very poor approximations.



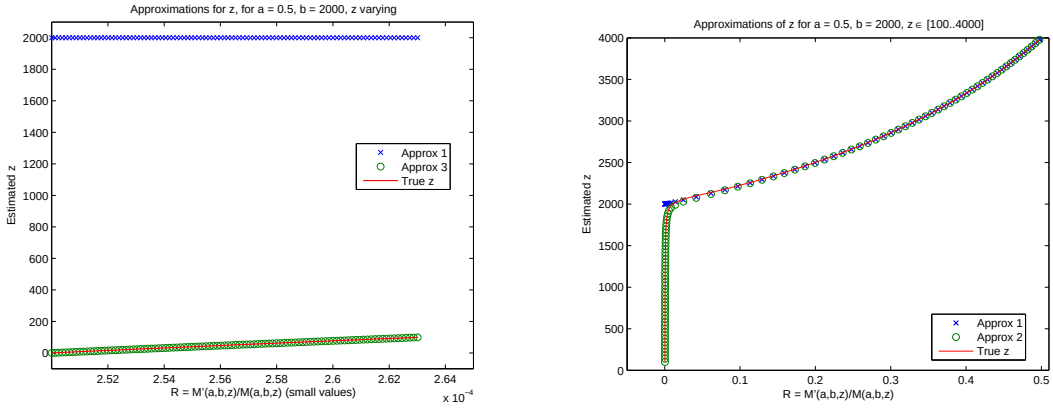
(a) (b)

Figure 6.4: Approximation for varying z with parameters a and b are held fixed at 5.5, and 2000, respectively. Subfigure (a) compares (A1) and (A2) for small values of $z \in [0.01..100]$. Subfigure (b) compares (A1) and (A2) for larger values of $z \in [100..4000]$. Notice that as $\bar{r} = {}_1F_1'/{}_1F_1$ increases, both approximations become accurate for large values of z. In fact, (A1) is more accurate than (A2) for larger values of z

Figure 6.6 shows approximations A1 and A3 for $a = 0.5$, as b is varied and z is held fixed. From the results above it may seem that approximations (A3) and (A4) perform similarly. A small attestation to this observation is provided by Figure 6.7 below.

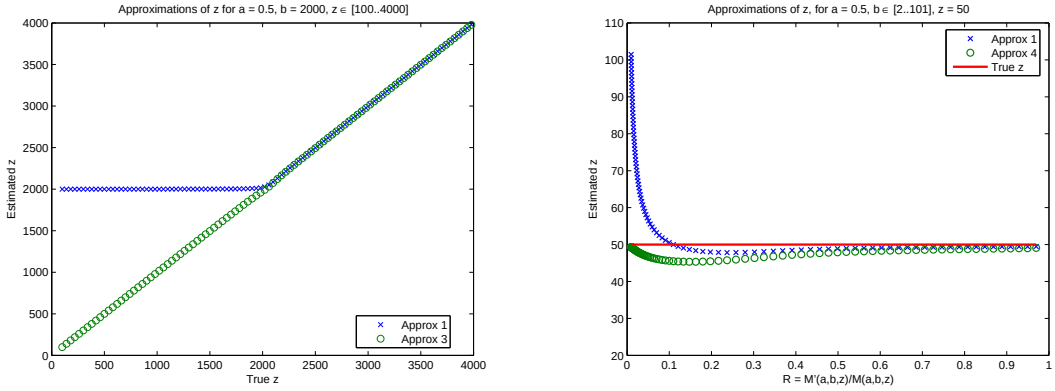
6.5 Clustering Algorithms

Given our derivation of EM algorithms for a mixture of vMF (moVMF) distributions, and a mixture of Watson (moW) distributions, we can now provide algorithms for clustering directional data.



(a) (b)

Figure 6.5: Approximation for varying z with with parameters a and b are held fixed at 0.5, and 2000.0, respectively. Subfigure (a) compares (A1) and (A2) for small values of $z \in [0.01..100]$. Subfigure (b) compares (A1) and (A2) for larger values of $z \in [100..4000]$. Notice that as $\bar{r} = {}_1F_1'/{}_1F_1$ increases, both approximations become accurate for large values of z. In fact, (A1) is more accurate than (A2) for larger values of z.



(a) (b)

Figure 6.6: Subfigure (a) shows the approximation of Figure 6.5(b) but with the true value of z as the x-axis. Subfigure (b) shows a plot of how approximation (A1) compares against (A4) as b is varied from 2 to 101. The true z was held constant at 50, and $a = 0.5$ was also fixed. We see that (A4) consistently outperforms (A1).

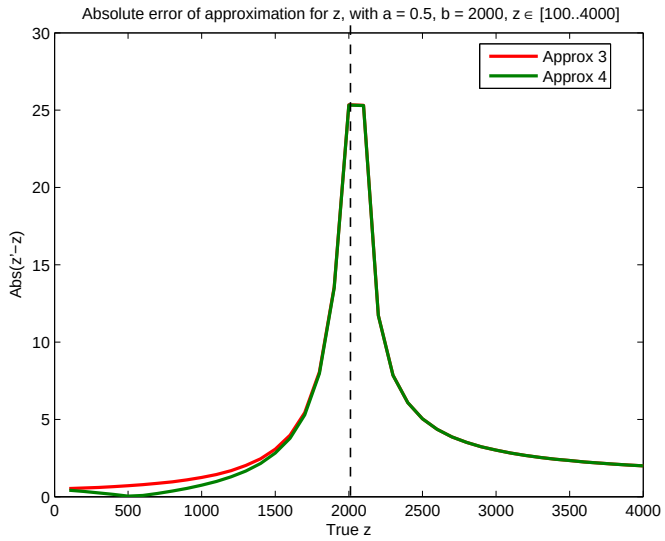


Figure 6.7: Absolute error of approximation of z for fixed $a = 0.5$, $b = 2000$, and $z \in [100..4000]$. After a point both (A3) and (A4) yield the same results. Note from the figure how the error shoots up when the true value of z is close to b. It is a known fact that asymptotic approximations for Hypergeometric functions break down when a is small, but both b and z are of comparable magnitude. Hence, some further scope of improvement is possible.

6.5.1 Algorithms for moVMF

The clustering algorithms for moVMF distributions are centered on soft and hard-assignment schemes and are titled `soft-moVMF` and `hard-moVMF` respectively. The `soft-moVMF` algorithm (Algorithm 6.10) estimates the parameters of the mixture model exactly following the derivations in Section 6.3.1 using EM. Hence, it assigns soft (or probabilistic) labels to each point that are given by the posterior probabilities of the components of the mixture conditioned on the point. On termination, the algorithm gives the parameters $\Theta = \{\alpha_k, \boldsymbol{\mu}_k, \kappa_k\}_{k=1}^{K}$ of the K vMF distributions that model the dataset $\boldsymbol{A}$, as well as the *soft-clustering*, i.e., the posterior probabilities $p(k|\boldsymbol{a}_i, \Theta)$, for all k and i (given by the β_{ik} values)

ALGORITHM 6.10: `soft-moVMF`

SOFTVMF($\boldsymbol{A}$)
Input: $\boldsymbol{A} \in \mathbb{S}^{M-1}$, K: number of clusters
Output: Soft clustering of $\boldsymbol{A}$ over a mixture of K vMF distributions
{Initialize}
$\alpha_k, \boldsymbol{\mu}_k, \kappa_k$ for $1 \leq k \leq K$
while not converged
  The E (Expectation) step of EM
  for $i = 1$ **to** N
    for $k = 1$ **to** K
      $p_k(\boldsymbol{a}_i|\boldsymbol{\theta}_k) \leftarrow c_M(\kappa_k) e^{\kappa_k \boldsymbol{\mu}_k^T \boldsymbol{x}_i}$
      $\beta_{ik} = p(k|\boldsymbol{a}_i, \boldsymbol{\Theta}) \leftarrow \dfrac{\alpha_k p_k(\boldsymbol{a}_i|\boldsymbol{\theta}_k)}{\sum_{l=1}^{K} \alpha_l p_l(\boldsymbol{a}_i|\boldsymbol{\theta}_l)}$
    end *for.*
  end *for.*
  The M (Maximization) step of EM
  for $k = 1$ **to** K
    $\alpha_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} \beta_{ik}$
    $\boldsymbol{\mu}_k \leftarrow \sum_{i=1}^{N} \boldsymbol{a}_i \beta_{ik}, \quad \bar{r} \leftarrow \|\boldsymbol{\mu}_k\|/(n\alpha_k), \quad \boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k/\|\boldsymbol{\mu}_k\|$
    $\kappa_k \leftarrow \frac{\bar{r}M - \bar{r}^3}{1-\bar{r}^2}$
  end *for.*
end *while.*

The `hard-moVMF` algorithm (Algorithm 6.11) estimates the parameters of the mixture model using a hard assignment, or, *winner takes all* strategy. In other words, we do the assignment of the points based on a derived posterior distribution, wherein the E-step that estimates β_{ik} is replaced by

$$\beta_{ik} \leftarrow \begin{cases} 1, & \text{if } \; k = \underset{k'}{\operatorname{argmax}} \; \alpha_{k'} \; p_{k'}(\boldsymbol{a}_i|\boldsymbol{\theta}_{k'}) \\ 0, & \text{otherwise.} \end{cases} \tag{6.29}$$

After these hard assignments, each point $\boldsymbol{a}_i$ *belongs* to a single cluster. Upon termination, Algorithm 6.11 yields a hard clustering of the data and the parameters $\boldsymbol{\Theta} = \{\alpha_k, \boldsymbol{\mu}_k, \kappa_k\}_{k=1}^{K}$.

ALGORITHM 6.11: `hard-moVMF`

HARDVMF($\boldsymbol{A}$)
Input: $\boldsymbol{A} \in \mathbb{S}^{M-1}$, K: number of clusters
Output: A disjoint K-partitioning of $\boldsymbol{A}$
{Initialize}
$\alpha_k, \boldsymbol{\mu}_k, \kappa_k$ for all $1 \leq k \leq K$
while not converged
  {The Hardened E-step of EM}
  for $i = 1$ **to** N
    for $k = 1$ **to** K
      $p_k(\boldsymbol{a}_i|\boldsymbol{\theta}_k) \leftarrow c_M(\kappa_k) e^{\kappa_k \boldsymbol{\mu}_k^T \boldsymbol{a}_i}$
      $\beta_{ik} \leftarrow \begin{cases} 1, & \text{if } \ k = \text{argmax}_{k'} \ p_{k'}(\boldsymbol{a}_i|\boldsymbol{\theta}_{k'}) \\ 0, & \text{otherwise.} \end{cases}$
    end *for*
  end *for*
  {The M-step of EM}
  Same as in Algorithm 6.10
end *while*.

6.5.2 Connection to Spherical Kmeans

In this section we show that upon enforcing certain restrictive assumptions on the generative model, the `spkmeans` algorithm (Algorithm 6.12) can be viewed as a special case of both the `soft-moVMF` and `hard-moVMF` algorithms.

More precisely, assume that in our mixture of vMFs, the priors of all the components are equal, i.e., $\alpha_k = 1/K$ for all k. Further assume that all the components have (equal) infinite concentration parameters, i.e., $\kappa_k = \kappa \to \infty$ for all k. Under these assumptions the E-step in the `soft-moVMF` algorithm reduces to assigning a point to its *nearest* cluster, where nearness is computed as a cosine similarity between the point and the cluster representative. Thus,

a point $\boldsymbol{a}_i$ will be assigned to cluster $k^* = \text{argmax}_k \ \boldsymbol{a}_i^T \boldsymbol{\mu}_h$, since

$$p(k^*|\boldsymbol{a}_i, \Theta) = \lim_{\kappa \to \infty} \frac{e^{\kappa \ \boldsymbol{x}_i^T \boldsymbol{\mu}_{k^*}}}{\sum_{k=1}^{K} e^{\kappa \ \boldsymbol{a}_i^T \boldsymbol{\mu}_k}} = 1,$$

and $p(k|\boldsymbol{a}_i, \Theta) \to 0$, as $\kappa \to \infty$ for all $k \neq k^*$.

To show that `spkmeans` can also be seen as a special case of the `hard-moVMF`, in addition to assuming the priors of the components to be equal, we further assume that the concentration parameters of all the components are equal, i.e., $\kappa_k = \kappa$ for all k. With these assumptions on the model, the estimation of the common concentration parameter becomes unessential since the hard assignment will depend only on the value of the cosine similarity $\boldsymbol{a}_i^T \boldsymbol{\mu}_h$, and `hard-moVMF` reduces to `spkmeans`.

ALGORITHM 6.12: `spkmeans`

SPKMEANS($\boldsymbol{A}$, K)
Input: $\boldsymbol{A} \in \mathbb{S}^{M-1}$
Output: A disjoint K-partitioning $\{A_k\}_{k=1}^{K}$ of $\boldsymbol{A}$
{Initialize}
$\boldsymbol{\mu}_k$ for $1 \leq k \leq K$
while not converged
  {The E (Expectation) step of EM}
  Set $A_k \leftarrow \emptyset$ for $1 \leq k \leq K$
  for $i = 1$ **to** N
    $A_k \leftarrow A_k \cup \{\boldsymbol{a}_i\}$ where $k = \text{argmax}_{k'} \ \boldsymbol{a}_i^T \boldsymbol{\mu}_{k'}$
  endfor
  {The M (Maximization) step of EM}
  for $k = 1$ **to** K
    $\boldsymbol{\mu}_k \leftarrow \dfrac{\sum_{\boldsymbol{a} \in A_k} \boldsymbol{a}}{\| \sum_{\boldsymbol{a} \in A_k} \boldsymbol{a} \|}$
  endfor
end*while.*

In our experiments, we also report results on `fskmeans` [10], an algorithm that lies within the same class, in the sense that like `spkmeans`, it can be derived from the mixture of vMF models with some restrictive assumptions. In `fskmeans`, the centroids of the mixture components are estimated as in `hard-movMF`. The κ value for a component is *explicitly set* to be inversely proportional to the number of points in the cluster corresponding to that component. This explicit choice simulates a frequency sensitive competitive learning that implicitly prevents the formation of null clusters, a well-known problem in regular kmeans [36].

6.5.3 Algorithms for moW

Just as for moVMF distributions, we can derive two clustering algorithms for moW distributions too. In Algorithms 6.10 and 6.11 we just need to replace $p_k(\boldsymbol{a}_i|\boldsymbol{\theta}_k)$ by the Watson density, and the M-steps need to be updated as per (6.14) and (A1–A4). We omit the details for brevity. We name the corresponding algorithms `hard-moW` and `soft-moW`, respectively.

6.5.4 Relation to diametric clustering

The Diametric Clustering algorithm of Dhillon et al. [84] groups together both correlated and anti-correlated data points. This amounts to grouping points while respecting axial symmetry. Immediately one might ask the question whether the diametric clustering procedure bears a relation to clustering based on mixtures of distributions that respect axial symmetry, i.e.,

distributions that essentially treat $\pm\boldsymbol{x}$ as the same. We answer this question in the affirmative, and show that the diametric clustering procedure of Dhillon et al. [84] is a limiting case of EM for a mixture of Watson distributions. To that end, first we recapitulate the diametric clustering procedure (taken from [84]) as Algorithm 6.13.

ALGORITHM 6.13: `diametric`

DIAMETRIC($\boldsymbol{A}$, K)
Input: $\boldsymbol{A} \in \mathbb{S}^{M-1}$
Output: A disjoint K-partitioning A_k of $\boldsymbol{A}$
{Initialize}
$\boldsymbol{\mu}_k$ for $1 \leq k \leq K$
while not converged
  {The E-step of EM}
  Set $A_k \leftarrow \emptyset$ for all $1 \leq k \leq K$
  for $i = 1$ **to** N
    $A_k \leftarrow A_k \cup \{\boldsymbol{a}_i\}$ where $k = \text{argmax}_{k'}(\boldsymbol{a}_i^T \boldsymbol{\mu}_{k'})^2$
  endfor
  {The M (Maximization) step of EM}
  for $k = 1$ **to** K
    $\boldsymbol{K}_k = [\boldsymbol{a}_i]$ such that $\boldsymbol{a}_i \in A_k$
    $\boldsymbol{\mu}_k \leftarrow \frac{\boldsymbol{K}_k \boldsymbol{\mu}_k}{\|\boldsymbol{K}_k \boldsymbol{\mu}_k\|}$
  endfor
end*while.*

From Algorithm 6.13 it is evident how it may be derived as a limiting case of EM for a mixture of Watson distributions. We can follow exactly the same arguments as in Section 6.5.2. The first view is based on a limiting view of `soft-moW` as $\kappa_k \to \infty$, because this sends $\beta_{ik} \to \{0, 1\}$. The second limiting view comes from ignoring kappas in `hard-moW`, by setting all of them to some fixed value κ^*. We omit the details for brevity.

6.6 Experimental Results

We now offer some experimental validation to assess the quality of clustering results achieved EM based clustering algorithms for moVMFs. Experimental results for moW have been reported by other researchers in the following paper [29], hence we do not replicate them here. Thus, for this section we compare the following four algorithms on numerous datasets.

1. Spherical K-Means [77]—`spkmeans`.
2. Frequency Sensitive Spherical K-Means [10]—`fskmeans`.
3. moVMF based clustering using hard assignments (§6.5.1)—`hard-moVMF`.
4. moVMF based clustering using soft assignments (§6.5.1)—`soft-moVMF`.

It has already been established that K-means using Euclidean distance performs much worse than `spkmeans` for text data [268], so we do not consider it here. Generative model based algorithms that use mixtures of Bernoulli or multinomial distributions, which have been shown to perform well for text datasets, have also not been included in the experiments. This exclusion is done because a recent empirical study over 15 text datasets showed the multinomial model to outperform simple versions of vMF mixture models (with κ constant for all clusters) for only one dataset (Classic3), and the Bernoulli model was inferior in all cases [300].

6.6.1 Datasets

The datasets that we used for empirical validation and comparison of our algorithms were carefully selected to represent some typical clustering problems. We also created various subsets of some of the datasets for gaining greater insight into the nature of clusters discovered or to model some particular clustering scenario (e.g. balanced clusters, skewed clusters, overlapping clusters etc.). We drew our data from five sources: Simulation, Classic3, Yahoo News, CMU 20 Newsgroup and Yeast Gene Expressions. For all the text document datasets, the toolkit MC [81] was used for creating a high-dimensional vector space model that each of the four algorithms utilized. MATLAB code was used to render the input as a vector space for both the simulated and gene-expression datasets.

- **Simulation.** We use simulated data to verify that the discrepancy between computed values of the parameters and their true values is small. Our simulated data serve the principal purpose of validating the "correctness" of our implementations. We used a slight modification of the algorithm given by Wood [288] to generate a set of data points following a given vMF distribution. We describe herein, two synthetic datasets. The first dataset **small-mix** is 2-dimensional and is used to illustrate soft-clustering. The second dataset **big-mix** is a high-dimensional dataset that could serve as a model for real world text datasets. Let the triple (n, d, k) denote the number of sample points, the dimensionality of a sample point and the number of clusters respectively.

1. **small-mix:** This data has $(n, d, k) = (50, 2, 2)$. The mean direction of each component is a random unit vector. Each component has $\kappa = 4$.

2. **big-mix:** This data has $(n, d, k) = (5000, 1000, 4)$. The mean direction of each component is a random unit vector, and the κ values of the components are 650.98, 266.83, 267.83, and 612.88. The mixing weights for each component are 0.251, 0.238, 0.252, and 0.259.

- **Classic3.** Classic3 is a well known collection of documents. It is an easy dataset to cluster since it contains documents from three well-separated sources. Moreover, the intrinsic clusters are largely balanced.

  1. **Classic3:** This corpus contains 3893 documents, among which 1400 CRANFIELD documents are from aeronautical system papers, 1033 MEDLINE documents are from medical journals, and 1460 CISI documents are from information retrieval papers. The particular vector space model used had a total of 4666 features (words). Thus, each document, after normalization, is represented as a unit vector in a 4666-dimensional space.

  2. **Classic300:** Classic300 is a subset of the Classic3 collection and has 300 documents. From each category of Classic3, we picked 100 documents at random to form this particular dataset. The dimensionality of the data was 5471.

3. **Classic400:** Classic400 is a subset of Classic3 that has 400 documents. This dataset has 100 randomly chosen documents from the MEDLINE and CISI categories and 200 randomly chosen documents from the CRANFIELD category. This dataset is specifically designed to create unbalanced clusters in an otherwise easily separable and balanced dataset. The dimensionality of the data was 6205.

- **Yahoo News (K-series).** This compilation has 2340 Yahoo news articles from 20 different categories. The underlying clusters in this dataset are highly skewed in terms of the number of documents per cluster, with sizes ranging from 9 to 494. The skewness presents additional challenges to clustering algorithms.

- **CMU Newsgroup.** The CMU Newsgroup dataset is a well known compilation of documents. We tested our algorithms on not only the original dataset, but on a variety of subsets with differing characteristics to explore and understand the behavior of our algorithms.

  1. **News20:** This standard dataset is a collection of 19,997 messages, gathered from 20 different USENET newsgroups. One thousand messages are drawn from the first 19 newsgroups, and 997 from the twentieth. The headers for each of the messages are then removed to avoid biasing the results. The particular vector space model used had 25924 words. News20 embodies the features characteristic of a typical text dataset—high-dimensionality, sparsity and significantly

overlapping clusters.

2. **Small-news20:** We formed this set by selecting 2000 messages from original News20 dataset. We selected 100 messages from each category in the original dataset. Hence this dataset has balanced classes (though there may be overlap). The dimensionality of the data was 13406.

3. **Same-100/1000** is a subset of the News20 dataset and comprises 100/1000 messages from 3 very similar newsgroups: comp.graphics, comp.os.ms-windows, comp.windows.x.

4. **Similar-100/1000** is a subset of the News20 dataset and comprises 100/1000 messages from 3 somewhat similar newsgroups: talk.politics.guns, talk.politics.mideast, talk.politics.misc.

5. **Different-100/1000** is a subset of the News20 dataset and comprises 100/1000 messages from 3 unrelated newsgroups: alt.atheism, rec.sport.baseball, sci.space.

- **Yeast Gene Expressions.** Gene-expression data was selected to offer a clustering domain different from text analysis. As previously motivated, the use of Pearson correlation for the analysis of gene expression data is common, so a directional model is well-suited. Coincident to this domain are the difficulties of cluster validation because of the unavailability of true labels. Such difficulties make the gene expression data a more challenging and perhaps a more rewarding domain for clustering.

Gene expression data is presented as a matrix of genes (rows) by expression values (columns). The expression vectors are constructed using DNA microarray experiments. We used a subset of the Rosetta Inpharmatics yeast gene expression set [136]. The original dataset consists of 300 experiments measuring expression of 6,048 yeast genes. Out of these we selected a subset of 996 genes for clustering. For each of the 996 genes the 300-element expression vector was normalized to have unit Euclidean norm.

6.6.2 Methodology

Except the gene expression dataset, performance of the algorithms on all the datasets has been analyzed using *mutual information* (MI) between the cluster and class labels. For gene data, due to the absence of true labels, we have to take recourse to reporting some internal figures of merit. We defer a discussion of the same to Section 6.6.7.

The MI gives the amount of statistical similarity between the cluster and class labels [58]. If X is a random variable for the cluster assignments and Y is a random variable for the pre-existing labels on the same data, then their MI is given by $I(X;Y) = E[\ln \frac{p(X,Y)}{p(X)p(Y)}]$ where the expectation is computed over the joint distribution of (X, Y) estimated from a particular clustering of the dataset under consideration. For `soft-moVMF` we "harden" the clustering produced by labeling a point with the cluster label for which it has the highest value of posterior probability (ties broken arbitrarily), in order to evaluate MI.

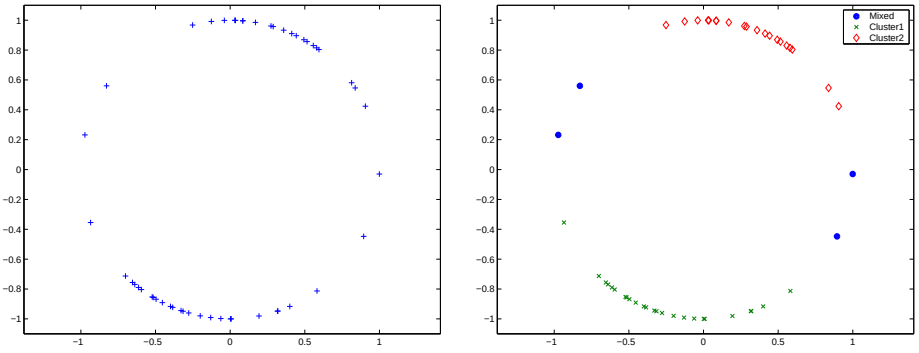
Note that variants of MI have been used to evaluate clustering algorithms by several researchers. [194] used a related concept called variation of information to compare clusterings. An MDL-based formulation that uses the MI between cluster assignments and class labels was proposed by [89].

All results reported herein have been averaged over 10 runs. All algorithms were started with the same random initialization to ensure fairness of comparison. Each run was started with a *different* random initialization. However, no algorithm was restarted within a given run and all of them were allowed to run to completion. Since the standard deviations of MI were reasonably small for all algorithms, to reduce clutter, we have chosen to omit a display of error bars in our plots.

6.6.3 Simulated Datasets

First, to build some intuition and confidence in the working of our vMF based algorithms we exhibit relevant details of `soft-moVMF`'s behavior on the small-mix dataset shown in Figure 6.8 (a).

The clustering produced by our soft cluster assignment algorithm is shown in Figure 6.8 (b). The points (taken clockwise) marked with solid circles have cluster labels $(0.15, 0.85)$, $(0.77, 0.23)$, $(.82, .18)$ and $(.11, .89)$, where a cluster label $(x, 1-x)$ for a point means that the point has probability x of belonging to Cluster 1 and probability $1-x$ of belonging to Cluster 2. All other points are categorized to belong to a single cluster by ignoring small (less than 0.10) probability values.



(a) The small-mix dataset. (b) A clustering of small-mix.

Figure 6.8: Small-mix dataset and its clustering by `soft-moVMF`.

The confusion matrix, obtained by "hardening" the clustering produced by `soft-moVMF` for the small-mix dataset is $\begin{bmatrix} 26 & 1 \\ 0 & 23 \end{bmatrix}$. As is evident from this confusion matrix, the clustering performed by `soft-moVMF` is excellent, though not surprising, since small-mix is a dataset with well-separated clusters. Further testimony to `soft-moVMF`'s performance is served by Table 6.2, which shows the discrepancy between true and estimated parameters for the small-mix collection. In the table $\boldsymbol{\mu}, \kappa, \alpha$ represent the true parameters and $\hat{\boldsymbol{\mu}}$,$\hat{\kappa}$,

Cluster	$\boldsymbol{\mu}$	$\hat{\boldsymbol{\mu}}$	κ	$\hat{\kappa}$	α	$\hat{\alpha}$
1	(-0.251, -0.968)	(-0.279, -0.960)	4	3.78	0.48	0.46
2	(0.399, 0.917)	(0.370, 0.929)	4	3.53	0.52	0.54

Table 6.2: True and estimated parameters for small-mix using `soft-moVMF`.

$\hat{\alpha}$ represent the estimated parameters. We can see that even in the presence of a limited number of data points in the small-mix dataset (50 points), the estimated parameters approximate the true parameters quite well.

Before moving onto real datasets let us briefly look at the behavior of

the algorithms on the larger dataset big-mix. On calculating MI as described previously we found that all the algorithms performed similarly with MI values close to one. We attribute this good performance of all the algorithms to the

$\min \boldsymbol{\mu}^T \hat{\boldsymbol{\mu}}$	$\text{avg } \boldsymbol{\mu}^T \hat{\boldsymbol{\mu}}$	$\max \frac{\lvert\kappa-\hat{\kappa}\rvert}{\lvert\kappa\rvert}$	$\text{avg } \frac{\lvert\kappa-\hat{\kappa}\rvert}{\lvert\kappa\rvert}$	$\max \frac{\lvert\alpha-\hat{\alpha}\rvert}{\lvert\alpha\rvert}$	$\text{avg } \frac{\lvert\alpha-\hat{\alpha}\rvert}{\lvert\alpha\rvert}$
0.994	0.998	0.006	0.004	0.002	0.001

Table 6.3: Performance of `soft-moVMF` on big-mix dataset.

availability of a sufficient number of data points and similar sized clusters. For reference Table 6.3 offers numerical evidence about the performance of `soft-moVMF` on the big-mix dataset.

6.6.4 Classic3 Family of Datasets

Table 6.4 shows typical confusion matrices obtained for the full Classic3 dataset. We observe that the performance of all the algorithms is quite similar and there is no added advantage yielded by using the general moVMF model as compared to the other algorithms. This observation can be explained by noting that the clusters of Classic3 are well separated and have a sufficient number of documents. For this clustering `hard-moVMF` yielded κ values of $(732.13, 809.53, 1000.04)$, while `soft-moVMF` reported κ values of $(731.55, 808.21, 1002.95)$.

`fskmeans`			`spkmeans`			`hard-moVMF`			`soft-moVMF`		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
1019	0	0	**1019**	0	0	**1018**	0	0	**1019**	0	1
1	6	**1386**	1	6	**1386**	2	6	**1387**	1	4	**1384**
13	**1454**	12	13	**1454**	12	13	**1454**	11	13	**1456**	13

Table 6.4: Comparative confusion matrices for 3 clusters of Classic3.

Table 6.5 shows the confusion matrices obtained for the Classic300 dataset. Even though Classic300 is well separated, the small number of documents per cluster makes the problem somewhat difficult for `fskmeans` and `spkmeans`, while `hard-moVMF` has a much better performance due to its model flexibility. The `soft-moVMF` algorithm performs appreciably better than the other three algorithms.

`fskmeans`			`spkmeans`			`hard-moVMF`			`soft-moVMF`		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
29	**38**	22	29	**38**	22	3	**72**	1	0	**98**	0
31	27	**38**	31	27	**38**	**62**	28	17	**99**	2	0
40	35	**40**	**40**	35	**40**	35	0	**82**	1	0	**100**

Table 6.5: Comparative confusion matrices for 3 clusters of Classic300.

`fskmeans`			`spkmeans`			`hard-moVMF`			`soft-moVMF`		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
27	16	**55**	27	17	**54**	**56**	28	20	0	0	**91**
51	**83**	12	**51**	**82**	12	44	**72**	14	**82**	**99**	2
23	1	**132**	23	1	**133**	1	0	**165**	19	1	**106**

Table 6.6: Comparative confusion matrices for 3 clusters of Classic400.

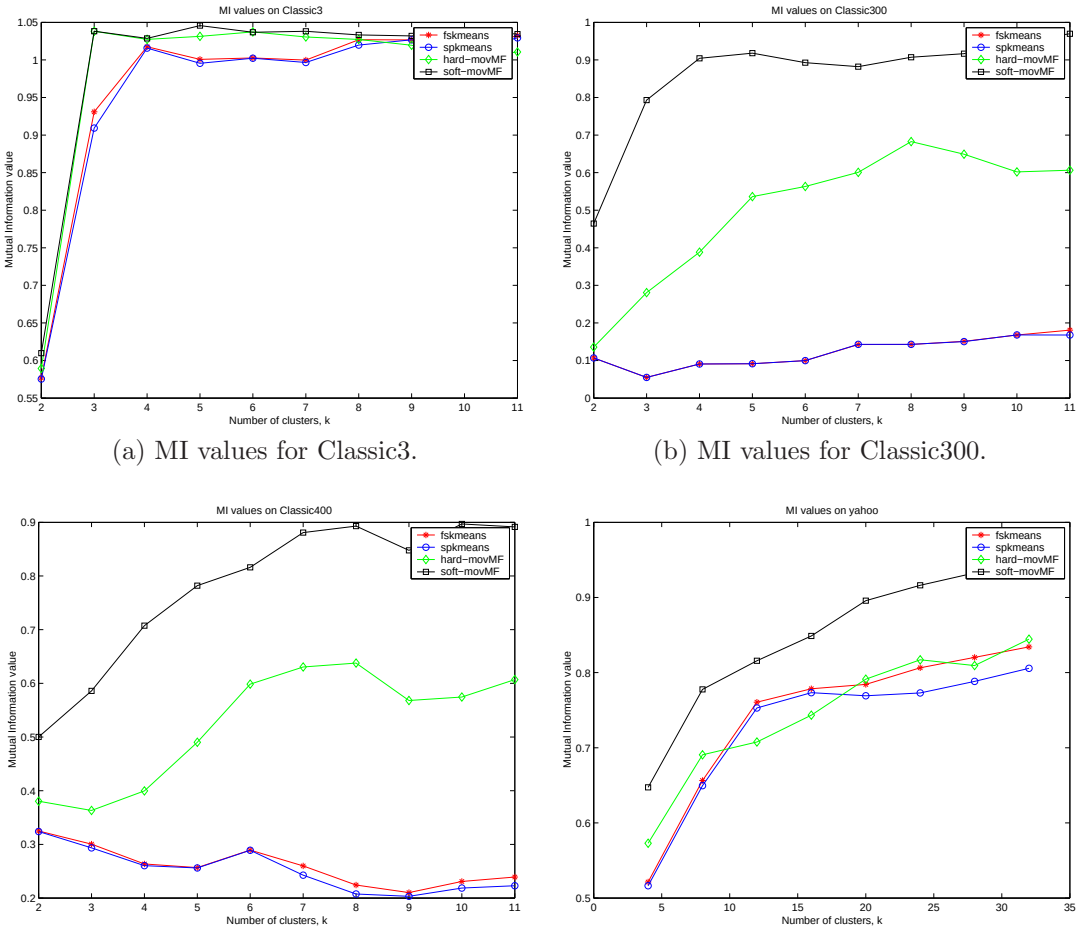
It seems that the low number of documents does not pose a problem for `soft-moVMF` and it ends up getting an almost perfect clustering for this dataset. Thus in this case, despite the low number of points per cluster, the superior modeling power of our moVMF based algorithms prevents them from getting trapped in inferior local-minima as compared to the other algorithms—resulting in a better clustering.

The confusion matrices obtained for the Classic400 dataset are displayed in Table 6.6. The behavior of the algorithms for this dataset is quite

interesting. As before, due to the small number of documents per cluster, `fskmeans` and `spkmeans` give a rather mixed confusion matrix. The `hard-moVMF` algorithm gets a significant part of the bigger cluster correctly and achieves some amount of separation between the two smaller clusters. The `soft-moVMF` algorithm exhibits a somewhat intriguing behavior. It splits the bigger cluster into two, relatively pure segments, and merges the smaller two into one cluster. When 4 clusters are requested from `soft-moVMF`, it returns 4 very pure clusters (not shown in the confusion matrices) two of which are almost equal sized segments of the bigger cluster.

An engaging insight into the working of the algorithms is provided by considering their clustering performance when they are requested to produce greater than the "natural" number of clusters. In Table 6.7 we show the confusion matrices resulting from 5 clusters of the Classic3 corpus. The matrices suggest that the moVMF algorithms have a tendency of trying to maintain larger clusters intact as long as possible, and breaking them into reasonably pure and comparably sized parts when they absolutely must. This behavior of our moVMF algorithms coupled with the observations in Table 6.6, suggest a clustering method in which one could generate a slightly higher number of clusters than required, and then agglomerate them appropriately.

The MI plots for the various Classic3 datasets are given in Figures 6.9(a)-(c). For the full Classic3 dataset (Figure 6.9(a)), all the algorithms perform almost similarly at the true number of clusters. However, as the number of clusters increases, `soft-moVMF` seems to outperform the others by a signifi-



(a) MI values for Classic3.

(b) MI values for Classic300.

(c) MI values for Classic400.

(d) MI values for Yahoo20.

Figure 6.9: Comparison of the algorithms for the Classic3 datasets and the Yahoo News dataset.

fskmeans			spkmeans			hard-moVMF			soft-moVMF		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
2	4	**312**	2	4	**323**	3	5	**292**	0	1	**1107**
8	**520**	10	8	**512**	9	**511**	1	0	5	**1455**	14
5	**936**	6	5	**944**	6	**514**	1	0	**526**	2	1
1018	0	1	**1018**	0	1	0	2	**1093**	**501**	0	0
0	0	**1069**	0	0	**1059**	5	**1451**	13	1	2	**276**

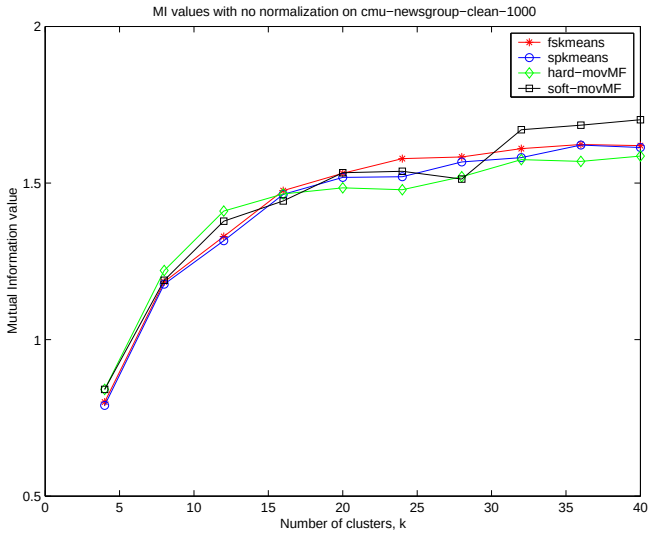
Table 6.7: Comparative confusion matrices for 5 clusters of Classic3.

cant margin. Another interesting point to note is that the MI values of the `hard-moVMF` algorithm fall very rapidly beyond the true number of clusters—this particular behavior could be used as a guideline to automatically figure out the "true" number of clusters for a dataset where such a number is unknown (e.g. gene data). For Classic300 (Figure 6.9(b)) and Classic400 (Figure 6.9(c)), `soft-moVMF` seems to significantly outperform the other algorithms. In fact, for these two datasets, `soft-moVMF` performs substantially better than the other three, even at the correct number of clusters. Among the other three, `hard-moVMF` seems to perform better than `spkmeans` and `fskmeans` across the range of clusters.

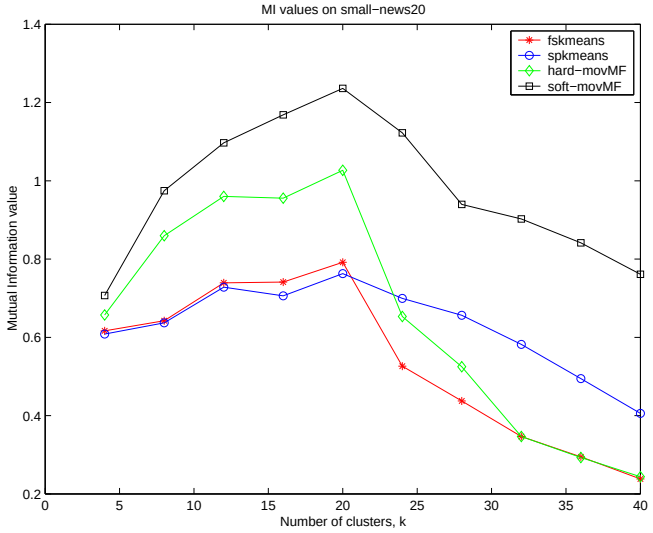
6.6.5 Yahoo News Dataset

The Yahoo News dataset is a relatively difficult dataset for clustering since it has a fair amount of overlap among its clusters and the number of points per cluster is low. In addition, the clusters are highly skewed in terms of their comparative sizes.

Results for the different algorithms can be seen in Figure 6.9(d). Over the entire range, `soft-moVMF` consistently performs better than the other algorithms. Even at the correct number of clusters $k = 20$, it performs significantly better than the other algorithms.

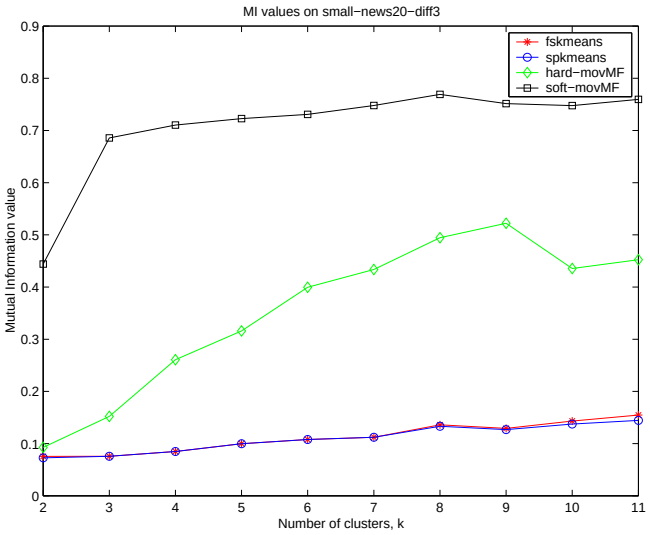


(a) MI values for News20.

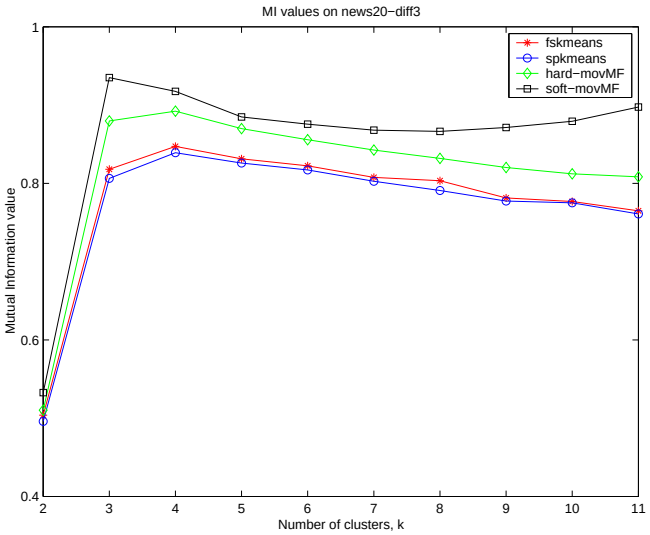


(b) MI values for Small-news20.

Figure 6.10: Comparison of the algorithms for the CMU Newsgroup and some subsets.

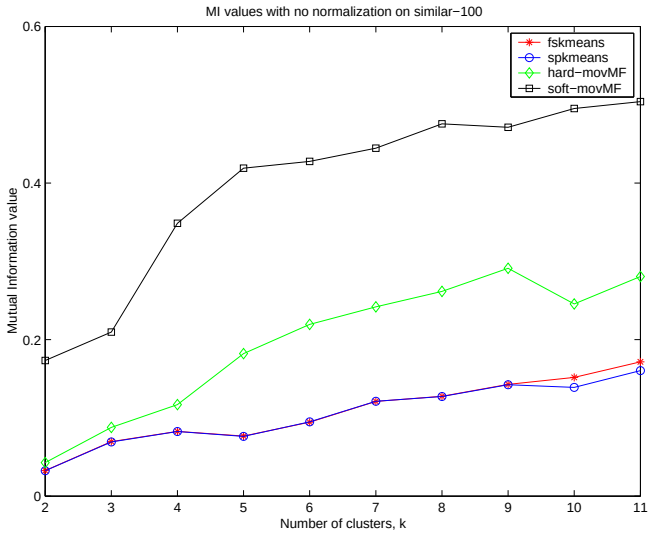


(c) MI values for Different-100.

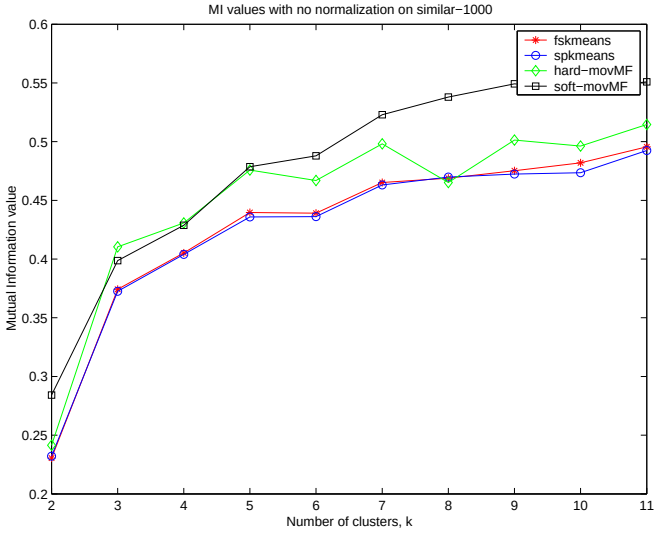


(d) MI values for Different-1000.

Figure 6.11: Comparison of the algorithms for the CMU Newsgroup and some subsets.



(e) MI values for Similar-100.



(f) MI values for Similar-1000.

Figure 6.12: Comparison of the algorithms for the CMU Newsgroup and some subsets.

6.6.6 CMU Newsgroup Family of Datasets

Now we discuss clustering performance of the four algorithms on the CMU Newsgroup datasets. Figure 6.10(a) shows the MI plots for the full News20 dataset. All the algorithms perform similarly until the true number of clusters after which `soft-moVMF` and `spkmeans` perform better than the others. We do not notice any interesting differences between the four algorithms from this Figure.

Figure 6.10(b) shows MI plots for the Small-News20 dataset and the results are of course different. Since the number of documents per cluster is small (100), as before `spkmeans` and `fskmeans` do not perform that well, even at the true number of clusters, whereas `soft-moVMF` performs considerably better than the others over the entire range. Again, `hard-moVMF` exhibits good MI values until the true number of clusters, after which it falls sharply. On the other hand, for the datasets that have a reasonably large number of documents per cluster, another kind of behavior is usually observed. All the algorithms perform quite similarly until the true number of clusters, after which `soft-moVMF` performs significantly better than the other three. This behavior can be observed in Figures 6.11(d), 6.12(f) and 6.13(b). We note that the other three algorithms perform quite similarly over the entire range of clusters. We also observe that for an easy dataset like Different-1000, the MI values peak at the true number of clusters, whereas for a more difficult dataset such as Similar-1000 the MI values increase as the clusters get further refined. This behavior is expected since the clusters in Similar-1000 are much

more overlapping than those in Different-1000.

6.6.7 Yeast Gene Expression Dataset

The gene dataset that we consider differs from text data in two major aspects. First, the data can have negative values, and second, we do not know the true labels for the data points.

Owing to the absence of true cluster labels for the data points, we evaluate the clusterings by computing certain internal figures of merit. These internal measures are commonly employed for evaluating clustering of genes [for e.g. 257]. Let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n\}$ be the set of data that is clustered into disjoint clusters $\mathcal{X}_1, \ldots, \mathcal{X}_k$. Let $\boldsymbol{\mu}_j$ denote the mean vector of the j-th cluster $(1 \leq j \leq k)$. The homogeneity of the clustering is measured by

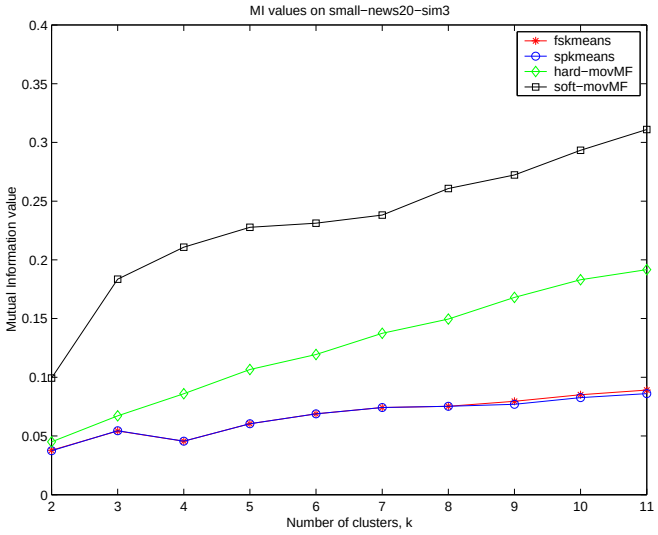
$$H_{avg} = \frac{1}{|\mathcal{X}|} \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in \mathcal{X}_j} \frac{\boldsymbol{x}^T \boldsymbol{\mu}_j}{\|\boldsymbol{x}\| \|\boldsymbol{\mu}_j\|}. \tag{6.30}$$

As can easily be seen, a higher homogeneity means that the individual elements of each cluster are quite similar to the cluster representative. We also take note of the minimum similarity

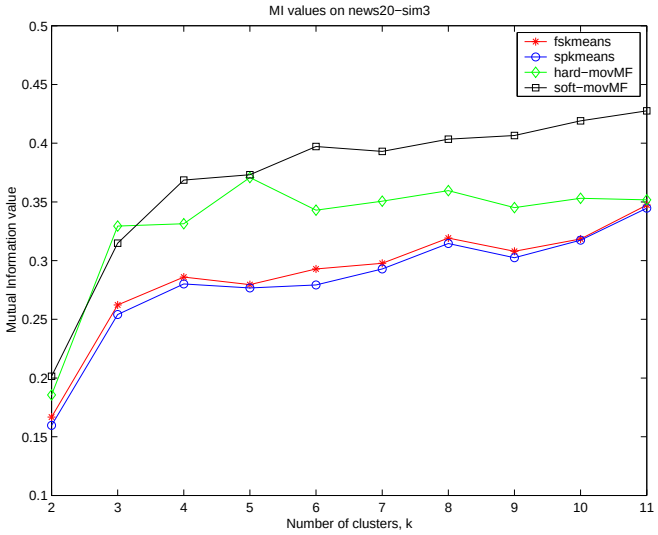
$$H_{min} = \min_{\substack{1 \leq j \leq k \\ \boldsymbol{x} \in \mathcal{X}_j}} \frac{\boldsymbol{x}^T \boldsymbol{\mu}_j}{\|\boldsymbol{x}\| \|\boldsymbol{\mu}_j\|}. \tag{6.31}$$

Both H_{avg} and H_{min} provide a measure of the intra-cluster similarity. We now define the inter cluster separation as

$$S_{avg} = \frac{1}{\sum_{i \neq j} |\mathcal{X}_i||\mathcal{X}_j|} \sum_{i \neq j} |\mathcal{X}_i \| \mathcal{X}_j| \frac{\boldsymbol{\mu}_i^T \boldsymbol{\mu}_j}{\|\boldsymbol{\mu}_i\| \|\boldsymbol{\mu}_j\|}. \tag{6.32}$$



(a) MI values for Same-100.



(b) MI values for Same-1000.

Figure 6.13: Comparison of the algorithms for more subsets of CMU Newsgroup data.

We also take note of the maximum inter-cluster similarity

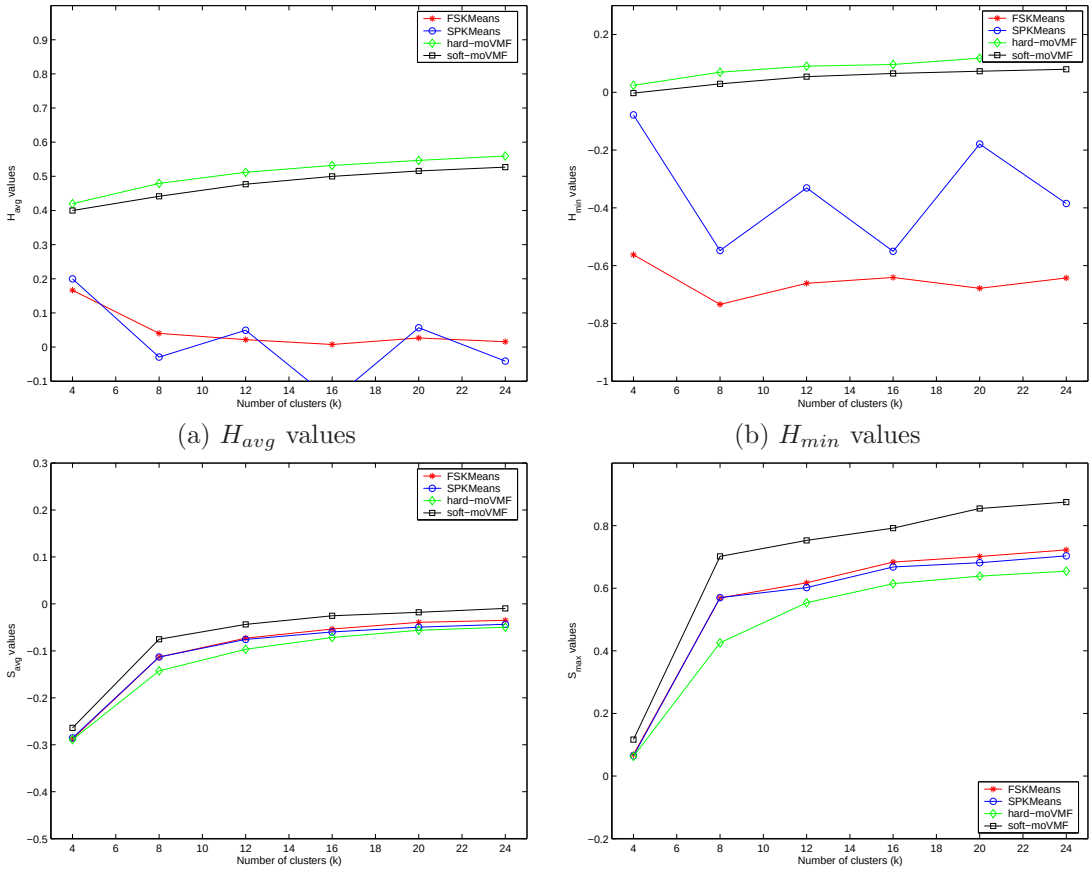
$$S_{max} = \max_{i \neq j} \frac{\boldsymbol{\mu}_i^T \boldsymbol{\mu}_j}{\|\boldsymbol{\mu}_i\| \|\boldsymbol{\mu}_j\|}. \tag{6.33}$$

It is easily seen that for a "good" clustering S_{avg} and S_{max} should be low.

Recently, researchers [175, 254] have started looking at supervised methods of evaluating the gene clustering results using public genome databases such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) and the gene ontology (GO). As of now, the evaluation techniques are still evolving and there is no consensus on how to best use the databases. For example, it is becoming clear that a pairwise precision-recall analysis of gene pairs may not be useful since the databases are currently incomplete due to lack of knowledge about all genes. Once the methods of evaluation become standardized, as a future work, we would like to evaluate the performance of our proposed algorithms with respect to the databases.

Figure 6.14 shows the various cluster quality figures of merit as computed for clusters of our gene expression data. A fact that one immediately observes is that `hard-moVMF` consistently performs better than all the other algorithms. This comes as somewhat of a surprise, because in almost all other datasets, `soft-moVMF` performs better (though, of course, the measures of evaluation are different for gene data as compared to the other datasets that we considered). Note that the figures of merit for `soft-moVMF` are computed after "hardening" the clustering results that it produced.

We see from Figure 6.14(a) that both `hard-moVMF` and `soft-moVMF`



(a) H_{avg} values

(b) H_{min} values

(c) S_{avg} values

(d) S_{max} values

Figure 6.14: Measures of cluster quality for gene data.

yield clusters that are much more homogeneous than those furnished by `fskmeans` and `spkmeans`. The inter-cluster similarities, as measured by S_{avg} and S_{max} are again the lowest for `hard-moVMF`, thereby indicating that `hard-moVMF` gives the best separated clusters of all the four algorithms. Though the inter-cluster similarities do not differ that much between the four algorithms, `soft-moVMF` seems to be forming clusters with higher inter-cluster similarity than other algorithms. We could explain this behavior of `soft-moVMF` by noting that

it tends to form overlapping clusters (because of soft-assignments) and those clusters remain closer even after hardening. Since H_{avg} essentially measures the average cosine similarity, we note that using our moVMF based algorithms, we are able to achieve clusters that are more coherent and better separated—a fact that could be attributed to the richer model employed by our algorithms. An inescapable observation is that our vMF based algorithms obtain a better average cosine similarity than `spkmeans`, implying that the richer vMF model allows them to escape the local minima that trap `spkmeans`.

6.6.8 Running time

This section shows a brief report of the running time differences between `hard-moVMF` and `soft-moVMF`. Table 6.8 shows these comparisons. These running time experiments were performed on an AMD Athlon based computer running the Linux operating system. From Table 6.8 we see that `hard-moVMF`

Clusters	Classic300	Classic3	News20
3	0.39s/11.56s	3.03s/109.87s	10.18s/619.68s
5	0.54s/17.99s	3.59s/163.09s	14.05s/874.13s
10	-	-	18.9s/1512s
20	-	-	29.08s/3368s

Table 6.8: Running time comparison between `hard-moVMF` and `soft-moVMF`. The times are indicated in the format "`hard-moVMF`/ `soft-moVMF`".

runs much faster than `soft-moVMF`, and this difference becomes even greater when the number of clusters desired becomes higher.

6.7 Discussion

From the experimental results, it seems that certain high-dimensional datasets, including text and gene-expression data, have properties that match well with the modeling assumptions of the vMF mixture model, and in certain cases with the Watson mixture model. This motivates further study of such models. For example, one can consider a hybrid algorithm that employs `soft-moVMF` for the first few (more important) iterations, and then switches to `hard-moVMF` for speed, and measure the speed-quality tradeoff that this hybrid provides. Another possible extension would be to consider an online version of the EM-based algorithms as discussed in this chapter, developed along the lines of Neal and Hinton [204]. Online algorithms are particularly attractive for dealing with streaming data when memory is limited, and for modeling mildly non-stationary data sources. We could also adapt a local search strategy such as the one in [82], for incremental EM to yield better local minima for both hard and soft-assignments.

The vMF and Watson distributions that we considered are amongst the simplest parametric distributions for directional data. There are more general of models on the unit sphere [187], such as the Bingham distribution, the Kent distribution, and the Fisher-Bingham distribution, the Pearson type VII distributions [187, 260, Figure 5.3], etc. that can potentially be more applicable in the general setting. For example, the Fisher-Bingham distributions have added modeling power since there are $O(M^2)$ parameters for each distribution. However, the parameter estimation problem, especially in high-dimensions, is

significantly more difficult for such models, as more parameters need to estimated from the data. Furthermore, we saw that the greatest difficulty towards parameter estimation was raised by the normalization constant. With increasingly complex directional distributions, the normalization constant becomes more and more complicated. For example, for the Bingham distribution, the normalization constant involves the matrix valued Hypergeometric function, which itself is difficult to compute. Thus, the generalization to more powerful models is not immediate, though it could be potentially useful.

6.8 Related Work

There has been an enormous amount of work on clustering a wide variety of datasets across multiple disciplines over the past fifty years [145]. The methods presented in this paper are tailored for high-dimensional data with directional characteristics, rather than for arbitrary datasets. In the learning community, perhaps the most widely studied high-dimensional directional data stem from text documents represented by vector space models. Much of the work in this domain uses discriminative approaches [267, 299]. For example, hierarchical agglomerative methods based on cosine, Jaccard or Dice coefficients were dominant for text clustering till the mid-1990s [235]. Over the past few years several new approaches, ranging from spectral partitioning [149, 299], to the use of generative models from the exponential family, e.g., mixture of multinomials or Bernoulli distributions [206] etc., have emerged. A fairly extensive list of references on generative approaches to text clustering

can be found in [301], who presented a unifying framework that captures the properties of virtually all generative approaches in this domain.

Of particular relevance to our work are `spkmeans` [77], and diametric clustering [84]. The former adapts the k-means algorithm to normalized data by using the cosine measure for cluster allocation and by re-normalizing the cluster means to unit length as well, while the latter treats anti-correlated points to be the same (essentially using squared cosine-similarity as its measure of proximity). Both of these algorithms appear to be discriminative at first glance, but turn out to be limiting cases of the generative models presented in this chapter.

The larger topic of clustering very high-dimensional data (dimension in the thousands or more), irrespective of whether it is directional or not, has also attracted great interest lately [76]. Again, most of the proposed methods to dealing with the curse of dimensionality in this context follow a density-based heuristic or a discriminatory approach [104]. Among generative approaches for clustering high-dimensional data, perhaps the most noteworthy is one that uses low dimensional projections of mixtures of Gaussians [62]. It turns out that one of our proposed methods alleviates problems associated with high dimensionality via an implicit local annealing behavior.

The vMF distribution is known in the literature on directional statistics [187], and the maximum likelihood estimates (MLE) of the parameters have been given for a single distribution. Recently Piater [226] obtained parameter estimates for a mixture for circular, i.e., 2-dimensional vMFs. In an

Appendix to his thesis, Piater [226] starts on an EM formulation for 2-D vMFs but cites the difficulty of parameter estimation (especially κ) and eventually avoids doing EM in favor of another numerical gradient descent based scheme. Mooney et al. [201] use a mixture of two circular von Mises distributions and estimate the parameters using a quasi-Newton procedure. Wallace and Dowe [282] perform mixture modeling for circular von Mises distributions and have produced a software called Snob that implements their ideas. McLachlan and Peel [193] discuss mixture analysis of directional data and mention the possibility of using Fisher distributions (3-dimensional vMFs), but instead use 3-dimensional Kent distributions [187]. They also mention work related to the clustering of directional data, but all the efforts included by them are restricted to 2-D or 3-D vMFs. Indeed, McLachlan and Peel [193] also draw attention to the difficulty of parameter estimation even for 3-D vMFs. Even for a single component, the maximum-likelihood estimate for κ involves inverting a ratio of two Bessel functions, and current ways of approximating this operation are inadequate for high-dimensional data. It turns out that our more accurate update for κ translates into a substantial improvement in the empirical results. The Watson distribution has already received attention elsewhere in the literature [29], where the authors followed our approach for moVMFs to obtain an efficient EM procedure. Our derivations in this chapter differ in terms of the approximation that we derive for the κ parameter for a moW. We remark that preliminary experiments show that the estimation error for the approximation to κ provided by Bijral et al. [29] can grow without bound—a situation that

does not seem to plague our approximations.

The connection between a generative model involving vMF distributions with constant κ and the `spkmeans` algorithm was first observed by Banerjee and Ghosh [10]. A variant that could adapt in an on-line fashion leading to balanced clustering solutions was developed by Banerjee and Ghosh [11]. Balancing was encouraged by taking a frequency-sensitive competitive learning approach in which the concentration of a mixture component was made proportional to the number of data points already allocated to it. Another online competitive learning scheme using vMF distributions for minimizing a KL-divergence based distortion was proposed by Sinkkonen and Kaski [261]. Note that the full EM solution was not obtained or employed in either of these works. Recently a detailed empirical study of several generative models for document clustering, including a simple mixture-of-vMFs model that constrains the concentration κ to be the same for all mixture components during any iteration was presented by Zhong and Ghosh [300]. Even with this restriction, this model was superior to both hard and soft versions of multivariate Bernoulli and multinomial models. These positive results further motivate the current paper in which we present the general EM solution for parameter estimation of a mixture of vMF distributions. This enhancement leads to even better clustering performance for difficult clustering tasks: when clusters overlap, when cluster sizes are skewed, and when cluster sizes are small relative to the dimensionality of the data. In the process, several new, key insights into the nature of hard vs. soft mixture modeling and the behavior of vMF

based mixture models are obtained. Of late, directional distributions seem to be attracting more interest. See for example the recent works [9, 29].

Chapter 7

Software and Implementation

In this chapter we describe some of the software that we have developed for implementing many of the algorithms derived as a part of this thesis. We include specific implementation details where relevant, to aid someone else trying to replicate our effort.

7.1 NNMA

Despite their simplicity, good software has been surprisingly lacking for even the simplest NNMA algorithms, for e.g., the original Lee & Seung algorithms. As a part of our research on NNMA algorithms, we have also implemented many of the derived algorithms. Our software is available both as a C++ implementation as well as through a MATLAB implementation. Figure 7.1 provides a screenshot of the MATLAB based interface to our NNMA software.

7.1.1 NNMA Algorithms

Our NNMA software includes implementations for solving NNMA problems with the following objective functions:

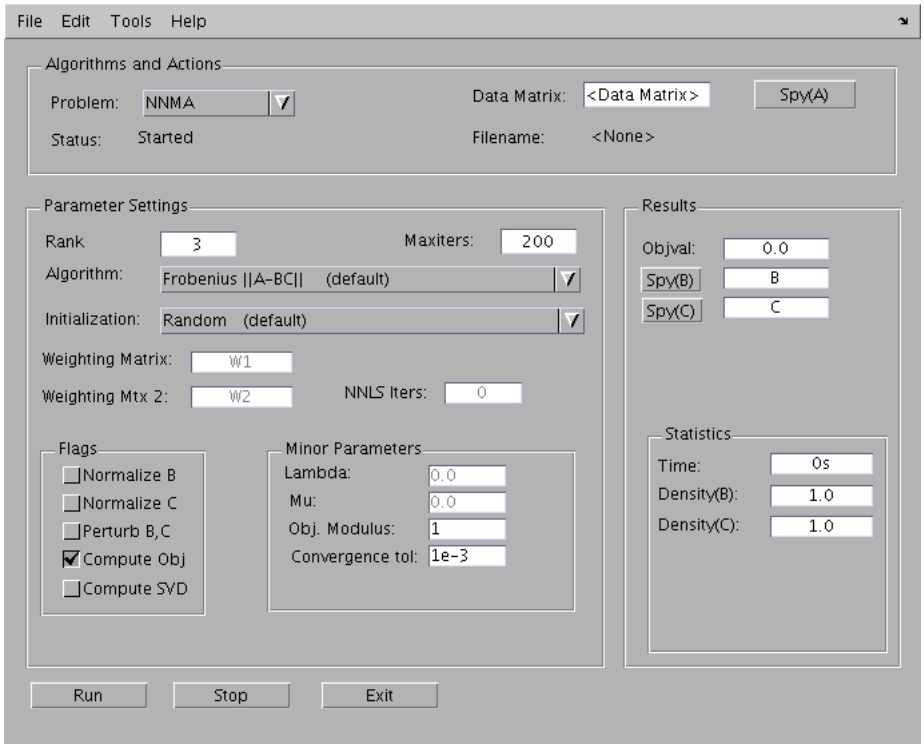


Figure 7.1: Screenshot of NNMA software

1. $\|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2$ via Lee & Seung's algorithm.

2. $\|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2$ via alternating non-negative least squares (NNLS)

3. $\|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2 + \lambda\|\boldsymbol{B}\|_{\mathrm{F}}^2 + \mu\|\boldsymbol{C}\|_{\mathrm{F}}^2$ via an adaptation of Lee & Seung's method.

4. $\|\boldsymbol{W} \odot (\boldsymbol{A} - \boldsymbol{BC})\|_{\mathrm{F}}^2$ via our weighted NNMA (§2.9.7)

5. $\|\boldsymbol{A} - \boldsymbol{W}_1\boldsymbol{BCW_2}\|_{\mathrm{F}}^2$ via our adaptation of Lee & Seung's method

6. $\|\boldsymbol{A} - \boldsymbol{RBC}\|_{\mathrm{F}}^2$ via our multi-factor generalization (§2.9.6) to NNMA

7. KL($\boldsymbol{A}\|\boldsymbol{BC}$) via Lee & Seung's algorithm

8. KL($\boldsymbol{A}\|\boldsymbol{BC}$) via Alternating KL-Divergence minimization

9. KL($\boldsymbol{A}\|\boldsymbol{BC}$) + $\lambda\|B\|_{\mathrm{F}}^2 + \mu\|C\|_{\mathrm{F}}^2$ via an adaptation of Lee & Seung's method

10. KL($\boldsymbol{A}\|\boldsymbol{RBC}$) via our multi-factor NNMA (§2.9.6)

11. KL($\boldsymbol{A}\|\boldsymbol{W}_1\boldsymbol{BCW}_2$) via our weighted NNMA (§2.9.7)

12. $D_\varphi(\boldsymbol{A};\boldsymbol{BC})$ via our generalized Bregman divergence method (via update (2.32))

Our software is implemented to work with both dense and sparse matrices as input, and uses efficient BLAS based linear algebra when possible.

Additional useful software

To supplement our NNMA software we have also developed wrapper libraries around SVDPACKC and ARPACK to permit the user to compute singular values and vectors for large sparse matrices. These wrapper libraries make it particularly easy to incorporate spectral computation capability into the users' own software, and are available via the author's website for public download.

```
./metric: -f FILE [-a [0-5]] [-h]

-f, --file=FILE      Filename containing non-metric matrix.
-a, --alg
  -a 0 ==> L_1 error metric nearness.
  -a 1 ==> L_2 error metric nearness.
  -a 2 ==> L_2 error with Bregman's algorithm.
  -a 3 ==> L_1 error, vectorized code
  -a 4 ==> L_2 error, Floyd-Warshall order of triangles
  -a 5 ==> KL  error, Bregman's algorithm
-h, --help           Display this help message
-m, --max=ITERS      Maximum iterations
-o, --out=[FILE]     Write output matrix to FILE (default = stdout)
-t, --tol=TOL        Convergence tolerance
-s, --scale=SCALE    Scale factor for epsilon in L1 code
--version            Display version number and exit.
```

Figure 7.2: Output of our metric nearness software

7.2 Metric Nearness

Our software for the Metric Nearness problem has contributed above and beyond its original intent. Metric Nearness is a constrained optimization problem with highly structured constraints, which we naturally exploit in our implementation. Our success with the metric nearness software prompted us to develop large-scale optimization software for linear and quadratic programs. That particular software effort lies outside the scope of this dissertation, but is available via the author's website.

Our metric nearness software is available both in a C++ as well as MATLAB implementation. Figure 7.2 shows the interface presented by our metric nearness software.

7.3 EM for vMF and Watson distributions

Usually implementing the EM algorithm is quite straightforward. However, for directional distributions such as vMF and Watson, the normalization constants are hard to compute. Furthermore, the maximum-likelihood parameter estimation requires the solution of difficult nonlinear equations, and for high-dimensional data, black-box approaches rapidly become infeasible. The implementation of EM for directional distributions also comes with engineering challenges such as overflows due to the explosive growth of the involved special functions. To tackle this problem, we had to implement our own versions, and our algorithms are described below in Section 7.3.1.

Our EM software for both mixtures of vMF and Watson distributions is available as part of this dissertation.

7.3.1 Elementary approaches to special function computation

Both the von Mises-Fisher density and Watson density require the computation of special functions. The former requiring computation of modified Bessel functions of the first kind (of real order and argument), i.e. $I_s(x)$, and the latter requiring computation of the confluent Hypergeometric function ${}_1F_1$ (with real arguments). Even though both of these functions have been very well studied in the literature and several implementations exist, they are insufficient for our applications for the following reasons:

1. The implementations are only for double precision arithmetic, leading to

overflow (or underflow) problems for high-dimensional data.

2. The implementations are available within commercial software such as Mathematica and Maple; this makes it infeasible for research code that we wish to provide to the community, because not everybody has access to these expensive software. Furthermore, for arguments typically encountered for high-dimensional data, the very general implementations within these software packages are too slow, making it impractical to run them on large scale data mining applications.

3. Specialized code written by other researchers is either technologically incompatible or does not scale well for our problems.

Thus, we had to resort to implementing these computations ourselves, using multi-precision arithmetic to deal with the problem of overflows. Naturally, our implementations are naïve, and could easily be improved significantly by incorporating asymptotic approximations or convergence acceleration (e.g., using Kummer's method, Aitken's acceleration, etc.) [35, 53, 253]. However, for our purposes a simple truncated power-series computation suffices and we present the resulting algorithms below.

Modified Bessel Function of the First Kind $I_s(x)$. We use the standard power-series representation (see [1])

$$I_s(x) = (x/2)^s \sum_{k \geq 0} \frac{(x^2/4)^k}{\Gamma(k+s+1)k!}. \tag{7.1}$$

Using the fact that $\Gamma(x+1) = x\Gamma(x)$, we rewrite (7.1) as

$$I_s(x) = \frac{(x/2)^s}{\Gamma(s)} \sum_{k\geq 0} \frac{(x^2/4)^k}{s(s+1)\cdots(s+k)k!}. \tag{7.2}$$

The form in (7.2) is amenable to a simple series computation because the ratio of the $(k+1)$-st term to the k-th term is

$$\frac{x^2}{4(k+1)(s+k+1)}. \tag{7.3}$$

Finally, to speed up the computation we use the following Stirling's approximation formula for the Gamma function (see [1, §6.1.37]):

$$\Gamma(x) \approx \left(\frac{x}{e}\right)^x \sqrt{\frac{2\pi}{x}}\left(1 + \frac{1}{12x} + \frac{1}{288x^2} - \frac{139}{51840x^3} + \cdots\right). \tag{7.4}$$

Thus we arrive at Algorithm 7.14 for approximating $I_s(x)$.

ALGORITHM 7.14: Computing $I_s(x)$ via truncated series

BESSELI(s,x)
Input: s, x: positive real numbers.
Output: $M \approx I_s(x)$
{Initialize}
$R \leftarrow 1.0, \quad \tau \leftarrow 10^{-10}, \quad t_1 \leftarrow \left(\frac{xe}{2s}\right)^s$
$t_2 \leftarrow 1 + \frac{1}{12x} + \frac{1}{288x^2} - \frac{139}{51840x^3}$
$t_1 \leftarrow t_1\sqrt{\frac{s}{2\pi}}/t_2$
$M \leftarrow 1/s$
while *not converged*
  $R \leftarrow R\frac{0.25s^2}{k(s+k)}$
  $M \leftarrow M + R$
  if $R/M < \tau$
    converged $\leftarrow$ *true*
  end *if*
end *while*
return M.

The Confluent Hypergeometric function ${}_1F_1(a,b,x)$**.** We use a truncated power series approximation for computing ${}_1F_1(a,b,x)$. The standard power series representation of ${}_1F_1$ is

$$ {}_1F_1(a,b,x) = \sum_{k\geq 0} \frac{a^{\overline{k}}}{b^{\overline{k}}} \frac{x^k}{k!}, \tag{7.5}$$

where $a^{\overline{k}} = a(a+1)(a+2)\cdots(a+k-1)$ denotes the *rising factorial* of a. The ratio of consecutive terms ($(k+1)$-st to k-th) is

$$ \frac{x(a+k)}{(k+1)(b+k)}. \tag{7.6}$$

Therefore, we have the following extremely simple algorithm for computing a truncated power series approximation (adapted from [203]).

ALGORITHM 7.15: Computing ${}_1F_1(a,b,x)$ via truncated series

KUMMER(a, b, x)
Input: a, b, x: positive real numbers.
Output: $M \approx {}_1F_1(a,b,x)$
{Initialize}
$M \leftarrow 1.0, \quad R \leftarrow 1.0, \quad i \leftarrow 0$
$\tau \leftarrow 10^{-10}$
while *not converged*
  $\beta \leftarrow \frac{(a+i)x}{(b+i)(i+1)}$
  $R \leftarrow \beta * R$
  $M \leftarrow M + R$
  $i \leftarrow i + 1$
  if $\beta < \tau$ **or** $M/R < \tau$
    converged $\leftarrow$ *true*
  end *if*
end *while*
return M.

Chapter 8

Conclusion

Although this may seem a paradox, all exact science is dominated by the idea of approximation.
– Bertrand Russell

In this thesis we developed several important data mining problems as matrix approximation or nearness problems. In particular, we obtained generalizations to basic problems such as non-negative matrix approximation (NNMA) (in Chapter 2) and other constrained low-rank matrix approximation problems of the form $\boldsymbol{A} \approx \boldsymbol{BC}$, where $\boldsymbol{B}$ and $\boldsymbol{C}$ are subject to linear inequality (in general convex) constraints (in Chapter 5). Our main algorithmic approach to solving these factored approximation problems was via the method of alternating minimization, wherein for difficult problems, we had to settle for just alternating descent.

A key convex optimization method that forms the basis of some of our implementations for solving nearness problems was motivated by our exploration of the metric nearness problem (Chapter 4). We solved metric nearness by developing efficient *triangle-fixing* algorithms, which themselves were efficient realizations of a more general row-action procedure. Subsequently, we created specialized row-action based algorithms for solving the convex subproblems arising in Chapter 5.

In this thesis, we additionally looked at a somewhat offbeat viewpoint of parametric mixture modeling, wherein we viewed the EM algorithm in the light of matrix nearness. We developed an efficient EM algorithm for modeling data using mixtures of directional distributions, in particular for mixtures of von Mises-Fisher and Watson distributions. This foray took us into numerical difficulties in estimating the parameters, especially due to the high-dimensionality of the data. We overcame the numerical challenges by developing efficient and accurate parameter approximations that led to significant computational savings.

Finally, to complement the theoretical discussion in the thesis, we presented a brief summary of the software that we have developed for solving the associated matrix approximation problems.

Appendices

Appendix A

Convex Analysis and Optimization

In this appendix we provide some brief background on convex analysis and optimization concepts that supplement the material in the thesis. We refer the reader to the following extremely useful books for more on convex analysis and optimization [26, 27, 33, 129, 183, 237].

A.1 Convex Sets and Functions

A set $C \subseteq \mathbb{R}^d$ is called *convex* if

$$\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in C,$$

whenever $\boldsymbol{x} \in C$, $\boldsymbol{y} \in C$ and $0 \leq \lambda \leq 1$. In other words, the set C contains all points on the line between any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in the set.

Convex sets satisfy some simple and useful properties, such as:

1. If C is convex, then so is αC, for all $\alpha \in \mathbb{R}$

2. The sum of two convex sets is convex, i.e., if C_1 and C_2 are two convex subsets of $\mathbb{R}^d$, then the set $C_1 + C_2 = \{\boldsymbol{x}_1 + \boldsymbol{x}_2 | \boldsymbol{x}_1 \in C_1, \boldsymbol{x}_2 \in C_2\}$ is convex.

3. A convex set remains convex under the action of a linear transformation.

4. The intersection of an arbitrary collection of convex sets is convex.

5. Given a finite set of points $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \in \mathbb{R}^d$, we can generate a corresponding convex set, called the *convex-hull* of S, which is defined as

$$\text{conv}(S) = \Big\{ \sum_i \lambda_i \boldsymbol{x}_i \quad | \boldsymbol{x}_i \in S, \lambda_i \geq 0, \text{and } \sum_i \lambda_i = 1 \Big\}.$$

A.1.1 Examples of convex sets

One of the most important class of convex sets that we use in this dissertation are halfspaces, i.e., sets of the form

$$C = \{\boldsymbol{x} \quad | \ \boldsymbol{a}^T \boldsymbol{x} \leq b, \text{where } \boldsymbol{a} \in \mathbb{R}^d, b \in \mathbb{R}\}.$$

Since the intersection of an arbitrary collection of convex sets is convex, it immediately follows that the set defined by the system of inequalities $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ is a convex set (with the convention that the empty set $\emptyset$ is also a convex set).

At this juncture we make the observation that the set of non-negative matrices is more than just a convex set, it is in fact a *convex cone*, i.e., a set K that is convex, and for each $\boldsymbol{x} \in K$, the vector $\alpha x \in K$ for all $\alpha \geq 0$. The set of distance matrices, as introduced in Chapter 4 forms a closed convex cone. We remark in passing that the set of positive semidefinite matrices also forms a closed convex cone (though not a polyhedral cone, since it is defined by an infinite number of inequalities).

A.1.2 Convex functions

We use the following definition (adapted from Theorem 4.1 of [237]) as our *definition* of a convex function.

Definition A.1 (Convex function)**.** Let $C \subseteq \mathbb{R}^d$ be a convex set, and f be a function from C to $(-\infty, +\infty]$. Then f is convex on C if and only if

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}), \qquad 0 < \lambda < 1.$$

One of the most useful theorems for determining whether a given function is convex or not is the following

Theorem A.2 ([237, Theorem 4.5])**.** *Let f be a twice continuously differentiable real-valued function on an open convex set $C \subseteq \mathbb{R}^d$. Then f is convex if and only if its Hessian matrix*

$$\boldsymbol{H} = [h_{ij}], \qquad h_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j},$$

is positive semi-definite for every $\boldsymbol{x} \in C$.

A.1.3 Bregman Divergences

Let S be a nonempty, open convex set such that its closure is contained in the domain of a function $\varphi : \Lambda \subseteq \mathbb{R}^n \to \mathbb{R}$. Also assume that $\varphi(x)$ has continuous first partial derivatives for all $x \in S$. From such a φ we construction a function $D_\varphi(x; y) : \bar{S} \times S \to \mathbb{R}$, where

$$D_\varphi(x; y) := \varphi(x) - \varphi(y) - \langle \nabla\varphi(y), x - y \rangle. \tag{A.1}$$

This function $D_\varphi(x; y)$ is called the Bregman divergence between x and y [45]. Under certain technical conditions φ is called a Bregman function [See 45, Def. 2.1.1]. For our purposes, we will always work with φs that are Bregman functions and following Censor and Zenios [45] we denote the family of Bregman functions $\mathscr{B}(S)$, where S is as described above.

Since our aim is to solve nearness problems, where we allow nearness to be measured by a Bregman divergence, we must first look at Bregman projections (or simply projections where the distance is measured by a Bregman divergence).

Definition A.3 (Bregman projection [45, Def 2.1.2])**.** Given $\Omega \subseteq \mathbb{R}^n$, $\varphi \in \mathscr{B}(S)$ and $y \in S$, a point $x^* \in \Omega \cap \bar{S}$ for which

$$P_\Omega(y) \equiv \min_{z \in \Omega \cap \bar{S}} D_\varphi(z; y) = D_\varphi(x^*; y),$$

is called the Bregman projection of the point y onto the set Ω.

Lemma A.4 ([45, Lemma 2.1.2])**.** *If $\varphi \in \mathscr{B}(S)$, then for any closed convex set $\Omega \subseteq \mathbb{R}^n$, such that the intersection of Ω with the closure of S is nonempty, i.e., $\Omega \cap \bar{S} \neq \emptyset$, and for any $y \in S$, there exists a unique Bregman projection $x^* \equiv P_\Omega(y)$.*

Now we come to the most important Lemma for our algorithms. The Lemma shows how to find the Bregman projection of a given point onto a hyperplane.

Lemma A.5 (Projection onto Hyperplane [45, Lemma 2.2.1]). *Let $\varphi \in \mathscr{B}(S)$, $H = \{\boldsymbol{x} | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}$, and assume that for every $y \in S$, the projection $P_H(y) \in S$. Then for any given $y \in S$, the system*

$$\begin{aligned} \nabla\varphi(x^*) &= \nabla\varphi(y) + \mu\boldsymbol{a} \\ \langle \boldsymbol{a}, x^* \rangle &= b, \end{aligned} \tag{A.2}$$

uniquely determines the point x^* that is the Bregman projection of y onto H. For a fixed H, the system also determines the parameter μ.

Before we can make immediate use of Lemma A.5 we look at one more useful Lemma that allows us to determine the sign of the parameter μ.

Lemma A.6 ([45, Lemma 2.2.2]). *Let $H = \{\boldsymbol{x} | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}$ and $\boldsymbol{y} \in S$. For any $\varphi \in \mathscr{B}(S)$ for which $P_H(\boldsymbol{y}) \in S, \forall \boldsymbol{y} \in S$, the parameter μ associated with $P_H(\boldsymbol{y})$ satisfies ($\boldsymbol{a} \neq \boldsymbol{0}$),*

$$\begin{aligned} \mu(b - \langle \boldsymbol{a}, \boldsymbol{y} \rangle) > 0, \quad & \text{if } \boldsymbol{y} \notin H, \\ \mu = 0, \quad & \text{if } \boldsymbol{y} \in H. \end{aligned}$$

We make use of this lemma for computing the generalized projections onto the half-space described by each triangle inequality for each different nearness measure. As a simple corollary, let us derive the orthogonal projection onto a hyperplane.

Corollary A.7 (Orthogonal Projection). *Let $\varphi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}\|^2$, where $\boldsymbol{x} \in \mathbb{R}^n$. Let $H = \{\boldsymbol{x} | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}$. Then $P_H(\boldsymbol{x})$ is given by,*

$$P_H(\boldsymbol{x}) = \boldsymbol{x} + \frac{1}{\|\boldsymbol{a}\|^2}[b - \langle \boldsymbol{a}, \boldsymbol{x} \rangle]^+ \boldsymbol{a}, \tag{A.3}$$

where $[\alpha]^+ = \alpha$ *if* $\alpha \geq 0$ *and* 0 *otherwise.*

Proof. From Lemmas A.5 and A.6 we know that $\boldsymbol{x}^* = P_H(\boldsymbol{x})$ must satisfy

$$\boldsymbol{x}^* - \boldsymbol{x}_0 = \boldsymbol{x} - \boldsymbol{x}_0 + \mu \boldsymbol{a}$$
$$\langle \boldsymbol{a}, \boldsymbol{x}^* \rangle = b.$$

Solving for μ and substituting, we immediately obtain (A.3). □

A.1.4 Bregman's Algorithm

Consider the problem,

$$\begin{aligned} \text{minimize} \quad & \varphi(x) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \bar{S} \end{aligned} \tag{A.4}$$

Bregman's algorithm (as described below) cycles through the constraints one by one performing an appropriate projection with correction to converge towards the optimal solution. The constraints are visited in a cyclic fashion and thus if the number of constraints is r, the $[kr] = r, k \in \mathbb{N}$ and $[n] = n \bmod r$ otherwise.

ALGORITHM A.16: Bregman's algorithm for Inequality Constrained Problems

BREGMAN(φ, $\boldsymbol{A}$, $\boldsymbol{b}$)
Input: φ, $\boldsymbol{A}$, $\boldsymbol{b}$ as in (A.4)
Output: $\operatorname{argmin} \varphi(x)$ s.t. constraints are satisfied

{Initialization}
$\boldsymbol{x}^0 \in \{\boldsymbol{x} \in S | \exists \boldsymbol{z} \in \mathbb{R}^m_+ \ni \nabla\varphi(\boldsymbol{x}) = -\boldsymbol{A}^T\boldsymbol{z}\}$ and $\boldsymbol{z}^0$ s.t. $\nabla\varphi(\boldsymbol{x}^0) = -\boldsymbol{A}^T\boldsymbol{z}^0$

{Iterative step} Given $\boldsymbol{x}^n$ and $\boldsymbol{z}^n$ perform the following updates
repeat
  $c_n := \min(\boldsymbol{z}^n_{[n]}, \mu_n)$
  $\nabla\varphi(\boldsymbol{x}^{n+1}) = \nabla\varphi(\boldsymbol{x}^n) + c_n\boldsymbol{a}_{[n]}$
  $\boldsymbol{z}^{n+1} = \boldsymbol{z}^n - c_n\boldsymbol{e}_{[n]}$
  {μ_n is calculated by solving (A.2) with $\boldsymbol{y} = \boldsymbol{x}^n$, $\boldsymbol{a} = \boldsymbol{a}_{[n]}$ and $b = \boldsymbol{b}_{[n]}$.}
until *convergence.*

There exist several decomposition procedures related to Bregman's method. For example Dykstra's method [34, 73] and Paul Tseng's extremely general and powerful dual block co-ordinate ascent procedure [277]. We refer the reader to [16, 17] for additional informative material related to Bregman-type decomposition methods.

Appendix B

Mathematical Background

In this appendix we provide some useful mathematical background that is useful for studying directional distributions (see Chapter 6).

B.1 Transformation to polar coordinates

Suppose we have an n vector $\boldsymbol{x}$ that we wish to translate to polar coordinates. We want to effect the transformation $\boldsymbol{x} = u(r, \boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{n-1})$ and $r = \|\boldsymbol{x}\|$. The co-ordinate transformation is generalized as below

$$\begin{aligned} x_k &\leftarrow r \sin\theta_1 \cdots \sin\theta_{k-1} \cos\theta_k, \qquad 1 \leq k < n, \\ x_n &\leftarrow r \sin\theta_1 \cdots \sin\theta_{n-1}. \end{aligned}$$

We can re-express this generalization in the following inductive way: If $z_1, \ldots, z_{n-1}$ are the co-ordinates in $n-1$ dimensional space and $x_1, \ldots, x_n$ are the co-ordinates in n-dimensional space, and $\boldsymbol{x} = u(\boldsymbol{\theta})$ is the transformation to polar

co-ordinates in $n-1$-space. Then we define (for $n > 3$):

$$\begin{aligned} x_i &= z_i \text{ for } 1 \leq i \leq n-2, \\ x_{n-1} &= z_{n-1} \cos\theta_{n-1}, \\ x_n &= z_{n-1} \sin\theta_{n-1}. \end{aligned}$$

The base case for the induction is $z_1 = r\cos\theta_1$ and $z_2 = r\sin\theta_1$. It is easy to verify that $\|\boldsymbol{x}\| = r$ and $\|\boldsymbol{z}\| = r$ as desired.

We know from vector calculus that: $d\boldsymbol{x} = |\det \boldsymbol{J}| d\boldsymbol{\theta}$ where $\boldsymbol{J}$ is the Jacobian matrix for the co-ordinate transformation and is given by

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x_1}{\partial r} & \frac{\partial x_1}{\partial \theta_1} & \cdots & \frac{\partial x_1}{\partial \theta_{n-1}} \\ \frac{\partial x_2}{\partial r} & \frac{\partial x_2}{\partial \theta_1} & \cdots & \frac{\partial x_2}{\partial \theta_{n-1}} \\ \frac{\partial x_3}{\partial r} & \vdots & \vdots & \frac{\partial x_3}{\partial \theta_{n-1}} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial x_n}{\partial r} & \frac{\partial x_n}{\partial \theta_1} & \cdots & \frac{\partial x_n}{\partial \theta_{n-1}} \end{bmatrix}.$$

The determinant of the Jacobian of the transformation is (where $s_i = \sin\theta_i$ and $c_i = \cos\theta_i$):

$$|\boldsymbol{J}| = r^{n-1} \begin{vmatrix} c_1 & -s_1 & 0 & \cdots & 0 \\ s_1 c_2 & c_1 c_2 & -s_1 s_2 & \cdots & 0 \\ s_1 s_2 c_3 & c_1 s_2 c_3 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \prod_{i=1}^{n-1} s_i & c_1 \prod_{i=2}^{n-1} s_i & \cdots & \cdots & c_{n-1} \prod_{i=1}^{n-2} s_i \end{vmatrix}. \tag{B.1}$$

To calculate this determinant let us first define the following:

$$D_n(k) = \begin{vmatrix} c_k & -s_k & 0 & \cdots & 0 \\ s_k c_{k+1} & c_k c_{k+1} & -s_k s_{k+1} & \cdots & 0 \\ s_k s_{k+1} c_{k+2} & c_k s_{k+1} c_{k+2} & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \prod_{i=k}^{n+k-2} s_i & c_k \prod_{i=k+1}^{n+k-2} s_i & \cdots & c_t \prod_{\substack{i=k \\ i \neq t}}^{n+k-2} s_i & c_{n+k-2} \prod_{i=k}^{n+k-3} s_i \end{vmatrix}.$$

Then it is clear that the determinant of the Jacobian (B.1) is given by $r^{n-1}D_n(1)$. Expanding $D_n(k)$ along c_k and $-s_k$ and we get the following:

$$\begin{aligned} D_n(k) = c_k & \begin{vmatrix} c_k c_{k+1} & -s_k s_{k+1} & \cdots & 0 \\ c_k s_{k+1} c_{k+2} & -s_k s_{k+1} s_{k+2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ c_k \prod_{i=k+1}^{n+k-2} s_i & \cdots & \cdots & c_{n+k-2} \prod_{i=k}^{n+k-3} s_i \end{vmatrix} \\ + \quad s_k & \begin{vmatrix} s_k c_{k+1} & -s_k s_{k+1} & \cdots & 0 \\ s_k s_{k+1} c_{k+2} & -s_k s_{k+1} s_{k+2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \prod_{i=k}^{n+k-2} s_i & \cdots & \cdots & c_{n+k-2} \prod_{i=k}^{n+k-3} s_i \end{vmatrix}. \end{aligned} \tag{B.2}$$

Taking out c_k common from the first term and s_k from the second term and noting the fact that $c_k^2 + s_k^2 = 1$ we see that (B.2) reduces to:

$$D_n(k) = \begin{vmatrix} c_{k+1} & -s_k s_{k+1} & \cdots & 0 \\ s_{k+1} c_{k+2} & -s_k s_{k+1} s_{k+2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \prod_{i=k+1}^{n+k-2} s_i & \cdots & \cdots & c_{n+k-2} \prod_{i=k}^{n+k-3} s_i \end{vmatrix}.$$

Since all but the first column contain an s_k, we factor it out yielding

$$D_n(k) = s_k^{n-2} D_{n-1}(k+1). \tag{B.3}$$

Also by direct evaluation we know that $D_3(j) = s_j$. Hence on iterating (B.3) we conclude that:

$$D_n(k) = s_k^{n-2} s_{k+1}^{n-3} \cdots s_{k+n-3} = \prod_{j=k+1}^{n+k-2} s_{j-1}^{n+k-1-j}.$$

Since we know that $|J| = r^{n-1}D_n(1)$ we get the following:

$$|J| = r^{n-1} \prod_{j=2}^{n} \sin^{n-j} \theta_{j-1}.$$

For our case we have $r = 1$ and hence we can write:

$$dx_1\, dx_2 \cdots dx_n = \prod_{j=2}^{n} \sin^{n-j} \theta_{j-1} d\theta_{j-1}.$$

At this point we would like to point out that the above elementary proof can be done in an algebraically less burdensome manner by using exterior derivatives. We point the interested reader to [202, Chapter 2] for more details.

B.2 Some integrals and functions

In this section we gloss over some functions and integrals that prove to be useful while studying directional distributions.

B.2.1 The Gamma Function

We state and prove a few properties about the Gamma function that are useful for understanding some of the derivations associated with vMF distributions.

Leonhard Euler was the first to obtain the following generalization of the factorial function (see [156])

$$n! = \lim_{m \to \infty} \frac{m^n m!}{(n+1)(n+2)\cdots(n+m)}.$$

A. M. Legendre introduced the notation: $n! = \Gamma(n+1) = n\Gamma(n)$ where

$$\Gamma(x) = \frac{x!}{x} = \lim_{m \to \infty} \frac{m^x m!}{x(x+1)(x+2)\cdots(x+m)}. \tag{B.4}$$

We now prove that

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1}\, dt.$$

Our proof is based upon Ex. 1.2.5-19 of [156]. We denote by $\Gamma_m(x)$ the quantity after the limit sign in equation (B.4) and we demonstrate that

$$\int_0^m \big(1 - t/m\big)^m t^{x-1}\, dt = \Gamma_m(x).$$

We can rewrite the integral above as (make the substitution $y = t/m$):

$$I_m(x) = \int_0^1 m^x (1-y)^m y^{x-1}\, dy.$$

Integrating by parts we find

$$\begin{aligned} I_m(x) &= m^x \left[\frac{(1-y)^m y^x}{x}\bigg|_0^1 + \frac{m}{x}\int_0^1 (1-y)^{m-1} y^x\, dy \right], \\ &= \frac{m^{x+1}}{x}\int_0^1 (1-y)^{m-1} y^x\, dy. \end{aligned}$$

We can see that if we inductively assume

$$\Gamma_{m-1}(x) = m^x \int_0^1 (1-y)^{m-1} y^{x-1}\, dy,$$

then we may write $xI_m(x) = \Gamma_{m-1}(x+1)$. Hence using induction we can show that $I_m(x) = \Gamma_m(x)$. Now we note the fact that as $m \to \infty$, $(1 - t/m)^m \to e^{-t}$. This limit enables us to write

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1}\, dt.$$

Note: The proof that $x\Gamma_m(x) = \Gamma_{m-1}(x+1)$ follows easily from the defintion of $\Gamma_m(x)$.

From either this integral or from the limiting definition we can verify the familiar property: $\Gamma(x+1) = x\Gamma(x)$. An important special case that comes up quite often is the value of $\Gamma(1/2)$. We shall evaluate it directly here. First we need a lemma:

Lemma B.1.

$$\int_0^\infty e^{-u^2}\,du = \sqrt{\pi}/2.$$

Proof. The integral in question is,

$$\begin{aligned} I &= \int_{-\infty}^{\infty} e^{-u^2}\,du, \\ I^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(u^2+v^2)}\,du\,dv. \end{aligned}$$

Now put $u = r\cos\theta$ and $v = r\sin\theta$. So we get,

$$\begin{aligned} I^2 &= \int_0^{2\pi} d\theta \int_0^\infty e^{-r^2} r\,dr, \\ &= \pi. \end{aligned}$$

Hence, $I = \sqrt{\pi}$. It is easy to see that the integral under consideration is just half of I hence the lemma is true. □ □

Now we calculate the value of $\Gamma(1/2)$.

$$\Gamma(1/2) = \int_0^\infty e^{-t} t^{-1/2}\,dt.$$

We substitute $t = u^2$ so that $dt = 2u\,du$. Hence we have:

$$\Gamma(1/2) = \int_0^\infty e^{-u^2} \frac{1}{u} 2u du = 2\int_0^\infty e^{-u^2}\,du.$$

But using Lemma B.1 we can conclude that: $\Gamma(1/2) = \sqrt{\pi}$.

B.2.2 The $\sin^n x$ integral

There are two ways that we show how to evaluate the following integral:

$$\int_0^\pi \sin^n x\, dx, \quad n > -1.$$

The first way is to type: `Integrate[Sin[x]^n,{x,0,π}]` in Mathematica and it will give back the answer:

$$\frac{\sqrt{\pi}\,\Gamma(\frac{1+n}{2})}{\Gamma(1+\frac{n}{2})}.$$

If one does not have recourse to Mathematica or some other such symbolic algebra system we could perform the integration as shown below.

$$\begin{aligned} I_n &= \int_0^\pi \sin^n x\, dx \\ &= \left[-\sin^{n-1} x \cos x\right]_0^\pi + (n-1)\int_0^\pi \sin^{n-2} x \cos^2 x\, dx \\ &= (n-1)\int_0^\pi \sin^{n-2} x\, dx - (n-1)\int_0^\pi \sin^n x\, dx \\ nI_n &= (n-1)I_{n-2}. \end{aligned}$$

By direct calculation we know that $I_2 = \pi/2$. So upon iteration we find out that

$$I_n = \frac{(n-1)(n-3)\cdots(n-2k+1)}{n(n-2)\cdots(n-2k)}\frac{\pi}{2} \quad ; \quad 2k = n-2.$$

We can write this as:

$$I_n = \frac{(\frac{n-1}{2})(\frac{n-3}{2})\cdots(\frac{n-2k+1}{2})}{\frac{n}{2}(\frac{n-2}{2})\cdots(\frac{n-2k}{2})}\frac{\pi}{2} \quad ; \quad 2k = n-2.$$

Since we know that $\Gamma(x+1) = x\Gamma(x)$ we can write

$$I_n = \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n+2}{2})}\frac{\pi}{2 \times \Gamma(3/2)}.$$

Using the fact that $\Gamma(3/2) = \sqrt{\pi}/2$ we conclude that

$$I_n = \frac{\sqrt{\pi}\,\Gamma(\frac{1+n}{2})}{\Gamma(1+\frac{n}{2})}.$$

I_n as given by the above equation is defined only for $n > -1$.

B.2.3 Useful formulae

The following differential equation gives rise to these modified Bessel functions:

$$z^2 w''(z) + z w'(z) - (z^2 + r^2) w(z) = 0. \tag{B.5}$$

This equation has solutions of the form: $w(z) = c_1 I_r(z) + c_z K_r(z)$ where $K_r(z)$ is the modified Bessel Function of the second kind.

The following two recurrence relations involving the derivative of the Bessel function are very useful in practice.

$$\kappa I_p'(\kappa) = p I_p(\kappa) + \kappa I_{p+1}(\kappa), \tag{B.6}$$

$$\kappa I_p'(\kappa) = \kappa I_{p-1}(\kappa) - p I_p(\kappa). \tag{B.7}$$

A standard definition of the modified Bessel function of order p and argument κ is

$$I_p(\kappa) = \sum_{r \geq 0} \frac{1}{\Gamma(p+r+1) r!} \left(\frac{\kappa}{2}\right)^{2r+p}. \tag{B.8}$$

Yet another definition is

$$I_p(\kappa) = \frac{2^{-p}\kappa^p}{\Gamma(p+1/2)\Gamma(1/2)} \int_0^\pi e^{\kappa \cos\theta} \sin^{2p}\theta d\theta, \tag{B.9}$$

which is equivalent to

$$I_p(\kappa) = \frac{1}{2\pi}\int_0^{2\pi} \cos p\theta e^{\kappa\cos\theta} d\theta. \tag{B.10}$$

Finally, we can also write the above is a form that might be suitable for numerical integration procedures as follows:

$$I_p(\kappa) = \frac{2^{-p}\kappa^p}{\Gamma(p+1/2)\Gamma(1/2)}\int_{-1}^{1} e^{\kappa t}(1-t^2)^{p-1/2} dt. \tag{B.11}$$

The following ratio is of principal importance to us,

$$A_p(\kappa) = \frac{I_{p/2}}{I_{p/2-1}}. \tag{B.12}$$

Another equivalent form can be derived using (B.11),

$$A_p(\kappa) = \frac{\int_{-1}^{1} t^2 e^{\kappa t}(1-t^2)^{(p-3)/2} dt}{\int_{-1}^{1} e^{\kappa t}(1-t^2)^{(p-3)/2} dt}. \tag{B.13}$$

We can obtain the following asymptotic representation for $A_p(\kappa)$,

$$\begin{aligned}\frac{\kappa}{p} - \frac{\kappa^3}{p^2\ (2+p)} + \frac{2\,\kappa^5}{p^3\ (2+p)\ (4+p)} + \frac{(-12-5\,p)\ \kappa^7}{p^4\,(2+p)^2\ (4+p)\ (6+p)} + \\ \frac{2\ (24+7\,p)\ \kappa^9}{p^5\,(2+p)^2\ (4+p)\ (6+p)\ (8+p)} + O(\kappa)^{10};\end{aligned} \tag{B.14}$$

in fact we can write $A_p(\kappa)$ as a convergent power series if $\kappa/p < 1$.

We are also interested in the derivative of $A_p(\kappa)$. We claim that the derivative of $A_p(\kappa)$, w.r.t. κ is given by,

$$A_p'(\kappa) = 1 - A_p(\kappa)^2 - \frac{p-1}{\kappa} A_p(\kappa). \tag{B.15}$$

Proof. Let $s = p/2 - 1$. Then we have,

$$A_p'(\kappa) = \frac{I_{s+1}'(\kappa)}{I_s(\kappa)} - \frac{I_{s+1}(\kappa)}{I_s(\kappa)}\frac{I_s'(\kappa)}{I_s(\kappa)}. \tag{B.16}$$

Now we make use of (B.7) to obtain

$$\frac{I_{s+1}'(\kappa)}{I_s(\kappa)} = 1 - \frac{s+1}{\kappa}\frac{I_{s+1}(\kappa)}{I_s(\kappa)}, \tag{B.17}$$

and we use (B.6) to write

$$\frac{I_s'(\kappa)}{I_s(\kappa)} = \frac{s}{\kappa} + \frac{I_{s+1}(\kappa)}{I_s(\kappa)}. \tag{B.18}$$

We know that $A_p(\kappa) = \frac{I_{s+1}(\kappa)}{I_s(\kappa)}$ hence we conclude that

$$A_p'(\kappa) = 1 - A_p(\kappa)^2 - \left(\frac{s}{\kappa} + \frac{s+1}{\kappa}\right) A_p(\kappa). \tag{B.19}$$

Putting in $p/2 - 1$ for s we get the desired conclusion. □ □

If we invert the power series representation for $A_p(\kappa)$ we obtain the following approximation for κ:

$$\begin{aligned} p\,\bar{R}\Bigg(1 + \frac{p\,\bar{R}^2}{2+p} + \frac{p^2\ (8+p)\ \bar{R}^4}{(2+p)^2\ (4+p)} + \frac{p^3\ (120 + p\ (14+p))\ \bar{R}^6}{(2+p)^3\ (4+p)\ (6+p)} + \\ \frac{p^4\ (24+p)\ (448 + p\ (112 + p\ (6+p)))\ \bar{R}^8}{(2+p)^4\,(4+p)^2\ (6+p)\ (8+p)}\Bigg). \end{aligned} \tag{B.20}$$

This estimate for κ does not really take into account the dimensionality of the data and thus for high dimensions (when κ is big by itself but κ/p is not very small or very big) it fails to yield accurate approximations. Note that $A_p(\kappa)$ is a ratio of Bessel functions that differ in their order by just one, so we can

use a well known continued fraction expansion for representing $A_p(\kappa)$. For notational simplicity let us write the continued fraction expansion as:

$$A_{2s+2}(\kappa) = \frac{I_{s+1}}{I_s} = \cfrac{1}{\frac{2(s+1)}{\kappa} + \cfrac{1}{\frac{2(s+2)}{\kappa} + \cdots}}. \tag{B.21}$$

The continued fraction on the right is well known [284]. Equation (B.21) and $A_p(\kappa) = \bar{R}$, allow us to write:

$$\frac{1}{\bar{R}} \approx \frac{2(s+1)}{\kappa} + \bar{R}.$$

Thus we can solve for κ to obtain the approximation,

$$\kappa \approx \frac{(2s+2)\bar{R}}{\bar{R} - \bar{R}^2}. \tag{B.22}$$

Since we made an approximation above, we incur some error, so we add a correction term (determined empirically) to the approximation of κ and obtain Equation (B.23),

$$\hat{\kappa} = \frac{\bar{R}p - \bar{R}^3}{1 - \bar{R}^2} \tag{B.23}$$

The above approximation can be generalized to include higher order terms in $\bar{R}$ to yield more accurate answers.[1] For $p = 2, 3$ highly accurate approximations can be found in [128]. In most cases this estimate for κ is good enough because as far as inference is concerned a very accurate estimate does not yield results commensurate with effort required to produce it. (The relative error between

[1]Note that if one really wants more accurate approximations, it is better to use (B.23) as a starting point and then perform a couple of Newton-Raphson iterations, because it is easy to evaluate $A'_p(\kappa) = 1 - A_p(\kappa)^2 - \frac{p-1}{\kappa} A_p(\kappa)$.

the true κ and this estimate has been found to be consistently lower than 0.05%)

For solving $A_p(\kappa) - \bar{R} = 0$, we can use (B.15) in any numerical method that may require the evaluation of the derivative of $A_p(\kappa)$ (such as Newton's method). In practice however, for very high dimensions (large p), Newton's iteration does not work that well because of numerical difficulties. In such cases, one could resort to other methods for improving the solution. Note that, again for practical purposes one does not really need very accurate calculations of κ. It is more of an academic numerical problem to find a good κ using an efficient and accurate root finder.

B.3 Hypergeometric functions

Hypergeometric functions provide one of the richest classes of functions in analysis. Traditionally the name "hypergeometric function" is used for Gauss' hypergeometric function

$$ {}_2F_1(a, b; c; z) = \sum_{k \geq 0} \frac{a^{\overline{k}} b^{\overline{k}}}{c^{\overline{k}} k!} z^k, \tag{B.24} $$

where $a^{\overline{k}}$ denotes the rising factorial (notation adopted from [109]), which is also denoted by the Pochhammer symbol $(a)_k = a(a+1)\dots(a+k-1)$. The generalized hypergeometric function ${}_pF_q$ is defined analogously as

$$ {}_pF_q = F(a_1, \dots, a_p; b_1, \dots, b_q; z) = \sum_{k \geq 0} \frac{a_1^{\overline{k}} \cdots a_p^{\overline{k}}}{b_1^{\overline{k}} \cdots b_q^{\overline{k}} k!} z^k. \tag{B.25} $$

The reader is referred to [1, 109] for more information on Hypergeometric functions. Several online resources also provide useful information.

For directional distributions, the hypergeometric function of interest is the confluent hypergeometric ${}_1F_1(a, b, z)$, also called *Kummer's* function. We now prove the two identities (2.33) and (2.56) that proved crucial to the derivation of our approximations.

Lemma B.2 (Derivative of ${}_1F_1$)**.**

$$\frac{d^n}{dz^n}\big({}_1F_1(a, b, z)\big) = \frac{a^{\overline{n}}}{b^{\overline{n}}}\,{}_1F_1(a + n, b + n, z).$$

Proof. We prove that $(d/dz)\,{}_1F_1(a, b, z) = (a/b)\,{}_1F_1(a + 1, b + 1, z)$, and the remainder of the proof follows by induction. We have

$$\begin{aligned}\frac{d}{dz}\,{}_1F_1(a, b, z) &= \sum_{k\geq 1} \frac{a^{\overline{k}}}{b^{\overline{k}}(k-1)!} z^{k-1} \\ &= \frac{a}{b} \sum_{k\geq 1} \frac{(a+1)^{\overline{k-1}}}{(b+1)^{\overline{k-1}}(k-1)!} z^{k-1} \\ &= \frac{a}{b} \sum_{k\geq 0} \frac{(a+1)^{\overline{k}}}{(b+1)^{\overline{k}}k!} z^{k}.\end{aligned}$$

□

Lemma B.3. *The following identity holds.*

$$(a + z)\,{}_1F_1(a + 1, b, z) + (b - a - 1)\,{}_1F_1(a, b, z) + (1 - b)\,{}_1F_1(a + 1, b - 1, z) = 0.$$

Proof. The sum of the coefficients of $\frac{a^{\overline{k}}}{b^{\overline{k}}k!}z^k$ from each of the three terms above is

$$(a + k) + \frac{k(b + k - 1)}{a} + b - (a + 1) - \frac{(a + k)(b + k - 1)}{a} = 0.$$

For obtaining the above sum of coefficients we used the easily proved identity $x^{\overline{m+n}} = x^{\overline{m}}(x+m)^{\overline{n}}$ (from which one easily concludes results such as $x^{\overline{k-1}} = x^{\overline{k}}/(x+k-1)$). □

Lemma B.4. *Given that $a > 0$ and $b > a$, the following identity holds.*

$$\frac{\Gamma(b-a)\Gamma(a)}{\Gamma(b)}\,{}_1F_1(a,b,z) = \int_0^1 e^{zt}t^{a-1}(1-t)^{b-a-1}dt, \tag{B.26}$$

Proof. Expand e^{zt} in its power series and integrate term by term to obtain the answer. We use the fact that (see [1] or [78] for basic facts about the $\Gamma(x)$ function)

$$\int_0^1 t^{x-1}(1-t)^{y-1}dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)},$$

along with the simple relation $a^{\overline{k}} = \Gamma(a+k)/\Gamma(a)$. □

B.4 Directional Distributions

The developments in this section are dependent upon the material presented in Sections B.1 and B.2. Following the treatment in [187], we will denote the probability element of $\boldsymbol{x}$ on a unit hyper-sphere by dS^{p-1}. The Jacobian of the transformation from (r,θ) to $\boldsymbol{x}$ is given by

$$dS^{p-1} = a_p(\theta)\,d\theta, \tag{B.27}$$

where we have (see Appendix A),

$$a_p(\theta) = \prod_{j=2}^{p-1} \sin^{p-j}\theta_{j-1}. \tag{B.28}$$

B.4.1 Uniform Distribution

If a direction $\boldsymbol{x}$ is uniformly distributed on S^{p-1} (unit hyper-sphere) then its probability element is $c_p\, dS^{p-1}$. The p.d.f. of θ is given by: $c_p a_p(\theta)$ (See Appendix A for a proof). Now we know that

$$\int c_p a_p(\theta) d\theta = 1, \tag{B.29}$$

hence using (B.28) we can write this as

$$c_p \int_0^{2\pi} d\theta_{p-1} \prod_{j=2}^{p-1} \int_0^{\pi} \sin^{p-j} \theta_{j-1} = 1. \tag{B.30}$$

Using the fact that (for $n > 0$, see Appendix A for a proof)

$$\int_0^{\pi} \sin^n x\, dx = \frac{\sqrt{\pi}\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n+2}{2})}, \tag{B.31}$$

we can easily solve equation (B.30) to give

$$c_p = \frac{\Gamma(p/2)}{2\pi^{p/2}}.$$

B.4.2 The von Mises-Fisher distribution

A unit random vector $\boldsymbol{x}$ is said to have $p-$variate *von Mises-Fisher* distribution if its p.e. is:

$$c_p(\kappa) e^{\kappa \boldsymbol{\mu}^T \boldsymbol{x}}\, dS^{p-1}, \quad \boldsymbol{x} \in S^{p-1} \subseteq \mathbb{R}^p. \tag{B.32}$$

Where $\|\boldsymbol{\mu}\| = 1$ and $\kappa \geq 0$. We will derive the value of c_p the normalizing constant using the fact that:

$$\int_{\boldsymbol{x} \in S^{p-1}} c_p(\kappa) e^{\kappa \boldsymbol{\mu}' \boldsymbol{x}}\, d\boldsymbol{x} = 1. \tag{B.33}$$

To evaluate the integral above we make the transformation $\boldsymbol{y} = \mathbf{Q}\boldsymbol{x}$, where $y_1 = \boldsymbol{\mu}^T\boldsymbol{x}$ and $\mathbf{Q}$ is an orthogonal transformation. $\boldsymbol{x} = \mathbf{Q}^{-1}\boldsymbol{y}$ so $d\boldsymbol{x} = |\frac{\partial}{\partial \boldsymbol{y}}\mathbf{Q}^{-1}\boldsymbol{y}|d\boldsymbol{y}$. But since $\mathbf{Q}$ is an orthogonal transformation we have $d\boldsymbol{x} = d\boldsymbol{y}$. It is easy to see that the first row of the matrix $\mathbf{Q}$ is $\boldsymbol{\mu}^T$. We now make the transformation to polar co-ordinates: $\boldsymbol{y} = u(\boldsymbol{\theta})$. Using Equations (B.27) and (B.28) we can rewrite the integral above as:

$$\int_0^{2\pi} d\theta_{p-1} \int_0^{\pi} e^{\kappa \cos\theta_1} \sin^{p-2}\theta_1 d\theta_1 \prod_{j=3}^{p-1} \int_0^{\pi} \sin^{p-j}\theta_{j-1} d\theta_{j-1}. \tag{B.34}$$

Using Eq. (B.31) we can rewrite the above integral as:

$$I = 2\pi \times J_1 \times \pi^{\frac{p-3}{2}} \frac{\Gamma(\frac{p-2}{2})}{\Gamma(\frac{p-1}{2})} \frac{\Gamma(\frac{p-3}{2})}{\Gamma(\frac{p-2}{2})} \cdots \frac{\Gamma(1)}{\Gamma(\frac{3}{2})}, \tag{B.35}$$

where J_1 is given by:

$$J_1 = \int_0^{\pi} e^{\kappa \cos\theta_1} \sin^{p-2}\theta_1 d\theta_1. \tag{B.36}$$

But we know from (B.9) that:

$$I_{\frac{p-2}{2}}(\kappa) = \left(\frac{\kappa}{2}\right)^{\frac{p-2}{2}} \frac{J_1}{\Gamma(\frac{p-1}{2})\Gamma(\frac{1}{2})}. \tag{B.37}$$

Hence on combining (B.35) and (B.37) and using the fact that $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ we see that the integral under question evaluates to:

$$I = \frac{(2\pi)^{p/2}}{\kappa^{p/2-1}} I_{p/2-1}(\kappa), \tag{B.38}$$

where $I_r(\kappa)$ is the modified Bessel Function as given by Eq. (B.9). We see that $c_p(\kappa) = I^{-1}$ and hence we have:

$$c_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}. \tag{B.39}$$

B.4.3 The Watson distribution

Now we derive the normalization constant for the Watson distribution over the unit hypersphere $\mathbb{S}^{d-1}$. The Watson density may be written as

$$W_d(\kappa, \boldsymbol{\mu}; \boldsymbol{x}) = c_d(\kappa) e^{\kappa(\boldsymbol{\mu}^T \boldsymbol{x})^2}, \quad \boldsymbol{x} \in \mathbb{S}^{d-1}, \kappa \in \mathbb{R}. \tag{B.40}$$

Normally, in the statistics literature this constant is computed relative to the uniform measure, which amounts to normalizing it by the volume of the unit hypersphere. Now, we make a change of variables to polar coordinates and integrate (B.40) over $\mathbb{S}^{d-1}$. Thus,

$$\begin{aligned}
\int_{\mathbb{S}^{d-1}} W_d(\kappa, \boldsymbol{\mu}; \boldsymbol{x}) d\boldsymbol{x} &= \int_{\mathbb{S}^{d-1}} c_d(\kappa) e^{\kappa(\boldsymbol{\mu}^T \boldsymbol{x})^2} = 1 \\
&= \int_0^{2\pi} d\theta_{d-1} \int_0^{\pi} e^{\kappa \cos^2 \theta_1} \sin^{d-2} \theta_1 d\theta_1 \prod_{j=3}^{p-1} \int_0^{\pi} \sin^{p-j} \theta_{j-1} d\theta_{j-1} \\
&= 2\pi \times I \times \pi^{\frac{d-3}{2}} \frac{1}{\Gamma(\frac{d-1}{2})}.
\end{aligned} \tag{B.41}$$

We now look at the integral denoted by I in the last step above. Let $\phi \leftarrow \theta - \frac{\pi}{2}$. Then we have

$$I = \int_0^{\pi} e^{\kappa \cos^2 \theta_1} \sin^{d-2} \theta_1 d\theta_1 = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{\kappa \sin^2 \theta_1} \cos^{d-2} \theta_1 d\theta_1.$$

The integrand above is an even function, hence we have

$$I = 2 \int_0^{\frac{\pi}{2}} e^{\kappa \sin^2 \theta_1} \cos^{d-2} \theta_1 d\theta_1.$$

Making the variable substitution $t \leftarrow \sin^2(\theta_1)$ and using (B.26), we can rewrite I as

$$I = \int_0^1 e^{\kappa t} t^{-\frac{1}{2}} (1-t)^{\frac{d}{2} - \frac{1}{2} - 1} dt = \frac{\Gamma(\frac{d}{2} - \frac{1}{2}) \Gamma(\frac{1}{2})}{\Gamma(\frac{d}{2})} {}_1F_1(\frac{1}{2}, \frac{d}{2}, \kappa).$$

Hence (B.41) becomes

$$2\pi^{\frac{d-1}{2}} \frac{\Gamma(1/2)}{\Gamma(d/2)} \, {}_1F_1(\frac{1}{2}, \frac{d}{2}, \kappa).$$

Hence $c_d(\kappa)$ is (using the fact that $\Gamma(1/2) = \sqrt{\pi}$)

$$\frac{\Gamma(d/2)}{2\pi^{d/2}} \, {}_1F_1(\frac{1}{2}, \frac{d}{2}, \kappa)^{-1}. \tag{B.42}$$

Normalizing (B.42) by the volume of the unit hypersphere, i.e., by $\Gamma(d/2)/(2\pi^{d/2})$ (see [78]) one obtains

$$c_d(\kappa) = \, {}_1F_1(\frac{1}{2}, \frac{d}{2}, \kappa)^{-1}, \tag{B.43}$$

since the volume of the unit hypersphere is for a proof of this latter fact).

Bibliography

[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publ. Inc., New York, June 1974. ISBN 0486612724.

[2] S. Amari. *Differential-Geometric methods in Statistics*. Springer, 1985.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. Assoc. Comput. Mach.*, 45:501–555, 1998.

[4] H. Asari. Nonnegative matrix factorization: a possible way to learn sound dictionaries. Preprint, 2005. URL `http://zadorlab.cshl.edu/asari/pdf/nmf.pdf`.

[5] L. Badea. Clustering and Metaclustering with Nonnegative Matrix Decompositions. In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, editors, *16th European Conference on Machine Learning*, Porto, Portugal, October 2005. Springer.

[6] L. Badea and D. Tilivea. Sparse factorizations of gene expressions guided by binding data. In *Pacific Symposium on Biocomputing*, volume 10, pages 447–458, 2005.

[7] D. Baier, W. Gaul, and M. Schader. Two-mode overlapping clustering with applications to simultaneous benefit segmentation and market structuring. In R. Klar and O. Opitz, editors, *Classification and Knowledge Organizations*, pages 557–566. Springer, Heidelberg, 1997.

[8] L. D. Baker and A. McCallum. Distributional clustering of words for text classification. In *ACM SIGIR*, pages 96–103. ACM, August 1998.

[9] A. Banerjee and S. Basu. Topic Models over Text Streams: A Study of Batch and Online Unsupervised Learning. In *SIAM Data Mining*, 2007. To appear.

[10] A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. In *Proceedings International Joint Conference on Neural Networks*, pages 1590–1595, May 2002.

[11] A. Banerjee and J. Ghosh. Frequency Sensitive Competitive Learning for Scalable Balanced Clustering on High-dimensional Hyperspheres. *IEEE Transactions on Neural Networks*, 15(3):702–719, May 2004.

[12] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Generative model-based clustering of directional data. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 19–28, 2003.

[13] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A Generalized Maximum Entropy Approach to Bregman Co-Clustering and Matrix Approximations. In *ACM SIGKDD International Conference on Knowledge Discovery and Datamining (KDD-2004)*, 2004.

[14] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman Divergences. In *SIAM International Conf. on Data Mining*, Lake Buena Vista, Florida, April 2004. SIAM.

[15] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman Divergences. *JMLR*, 6(6):1705–1749, October 2005.

[16] H. H. Bauschke and J. M. Borwein. Legendre functions and the method of random Bregman projections. *Journal of Convex Analysis*, 4:27–67, 1997.

[17] H. H. Bauschke and A. S. Lewis. Dykstra's algorithm with Bregman projections: a convergence proof. *Optimization*, 48:409–427, 2000.

[18] S. Behnke. Hebbian learning and competition in the Neural Association Pyramid. In *IJCNN*, Washington, DC, 1999.

[19] S. Behnke. Discovering hierarchical speech features using convolutional nonnegative matrix factorization. In *International Joint Conference on Neural Networks*, volume 4, pages 2758–2763, Portland, OR, 2003.

[20] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[21] A. Ben-Dor, B. Chor, R. M. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order preserving submatrix problem. In *6th International Conference of Computational Biology (RECOMB'02)*, pages 49–57, 2002.

[22] A. Berman and R. J. Plemmons. Eight types of matrix monotonicity. *Linear Algebra and its Applications*, 13:115–123, 1976.

[23] M. Berry, M. Browne, A. Langville, P. Pauca, and R. J. Plemmons. Algorithms and applications for approximation nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 2006. Preprint.

[24] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[25] M.W. Berry and M. Browne. Email Surveillance Using Nonnegative Matrix Factorization. In *Workshop on Link Analysis, Counterterrorism and Security, SIAM Data Mining*, pages 45–54, April 2005.

[26] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[27] D. P. Bertsekas, A. N., and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, April 2003. ISBN 1-886529-45-0.

[28] N. Le Bihan and G. Ginolhac. Three-mode data set analysis using higher order subspace method: application to sonar and seismo-acoustic signal processing. *Signal Processing*, 84:919–942, 2004.

[29] A. Bijral, M. Breitenbach, and G. Z. Grudic. Mixture of Watson Distributions: A Generative Model for Hyperspherical Embeddings. In *AISTATS*, 2007.

[30] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[31] D. Böhning. A review of reliable maximum likelihood algorithms for semi-parametric mixture models. *J. of Stat. Planning and Inference*, 47: 5–28, 1995.

[32] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 1999.

[33] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

[34] J. P. Boyle and R. L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In R. L. Dykstra, T. Robertson, and F. T. Wright, editors, *Advances in Order Restricted Statistical Inference*, volume 37 of *Lecture Notes in Statistics*, pages 28–47. Springer-Verlag, Iowa City, 1985.

[35] D. M. Bradley. A class of series acceleration formulae for the Catalan's constant. *The Ramanujan Journal*, 3(2):159–173, 1999.

[36] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, Microsoft Research, May 2000.

[37] L. M. Bregman. The relaxation method of finding the common point of convex sets and its applications to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[38] L. M. Bregman, Y. Censor, and S. Reich. Dykstra's Algorithm as the Nonlinear Extension of Bregman's Optimization Method. *Journal of Convex Analysis*, 6(2):319–333, 1999.

[39] J. Brickell, I. S. Dhillon, S. Sra, and J. A. Tropp. The Metric Nearness Problem. *SIAM J. Matrix Analysis and Appl.*, 2006. Submitted.

[40] R. Bro and S. de Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11:393–401, 1997.

[41] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *PNAS*, 101 (12):4164–4169, 2004.

[42] S. Busygin, G. Jacobsen, and E. Kramer. Double conjugated clustering applied to leukemia microarray data. In *SDM Workshop on Clustering High Dimensional Data*, 2002.

[43] S. L. Campbell and G. D. Poole. Computing nonnegative rank factorizations. *Linear Algebra and its Applications*, 35:175–182, 1981. ISSN 0024-3795.

[44] J. F. Cardoso. Multidimensional independent component analysis. In *Proc. ICASSP'98*, Seattle, WA, 1998.

[45] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[46] J-C. Chen. The nonnegative rank factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 62:207–217, 1984. ISSN 0024-3795.

[47] Z. Chen, A. Cichocki, and T. M. Rutkowski. Constrained Non-Negative Matrix Factorization Method for EEG Analysis in Early Detection of Alzheimer's Disease. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2006*, Toulouse, France, 2006.

[48] Y. Cheng and G. M. Church. Biclustering of Expression Data. In *8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, La Jolla, California, August 2000.

[49] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum Sum Squared Residue based Co-clustering of Gene Expression data. In *Proc. 4th SIAM International Conference on Data Mining (SDM)*, pages 114–125, Florida, 2004. SIAM.

[50] A. Cichocki, S. Amari, R. Zdunek, Z. He, and R. Kompass. Extended SMART Algorithms for Non-Negative Matrix Factorization. In *Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC*, Zakopane, Poland, 2006.

[51] A. Cichocki, R. Zdunek, and S. Amari. Csiszar's Divergences for Non-Negative Matrix Factorization: Family of New Algorithms. In *6th International Conference on Independent Component Analysis and Blind Signal Separation*, volume Springer LNCS 3889, pages 32–39, Charleston SC, USA, 2006.

[52] A. Cichocki, R. Zdunek, and S. Amari. New Algorithms for Non-Negative Matrix Factorization in Applications to Blind Source Separation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France, 2006.

[53] H. Cohen, F. Rodgriguez, and D. Zagier. Convergence acceleration of alternating series. *Experimental Mathematics*, 9(3), 2000.

[54] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaBoost, and Bregman distances. In *Thirteenth annual conference on COLT*, 2000.

[55] M. Collins, S. Dasgupta, and R. E. Schapire. A Generalization of Principal Components Analysis to the Exponential Family. In *NIPS 2001*, 2001.

[56] M. Cooper and J. Foote. Summarizing video using nonnegative similarity matrix factorization. In *IEEE Multimedia Signal Processing Workshop*, St. Thomas, USVI, December 2002.

[57] Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001. URL `http://mitpress.mit.edu/algorithms/`.

[58] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[59] L. Cranias, H. Papageorgiou, and S. Piperidis. Clustering: a technique for search space reduction in example-based machine translation. In *Proceedings 1994 IEEE ICSMC*, volume Humans, Information and Technology I, 1994.

[60] C. Cryer. The LU-factorization of totally positive matrices. *Linear Algebra and its Applications*, 7:83–92, 1973.

[61] I. Csiszar and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics and Decisions*, 1:205–237, 1984.

[62] S. Dasgupta. Learning mixtures of Gaussians. In *IEEE Symposium on Foundations of Computer Science*, 1999.

[63] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R.G. Lanckriet. A direct formulation for sparse pca using semidefinite programming.

In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 41–48. MIT Press, Cambridge, MA, 2005.

[64] J. Dattorro. *Euclidean Distance Geometry via Convex Optimization*. PhD thesis, Department of Electrical Engineering, Stanford University, June 2005.

[65] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic Metric Learning. In *ICML*, June 2007. Accepted.

[66] W. H. E. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of Mathematical Biology*, 49(4):461–467, 1987.

[67] W. F. de la Vega and C. Kenyon. A randomized approximation scheme for Metric MAX-CUT. *J. Comput. Sys. and Sci.*, 63:531–541, 2001.

[68] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[69] V. de Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal of Matrix Analysis and Applications*, 2006.

[70] A. Dekel and M. J. West. On percolation as a cosmological test. *The Astrophysical Journal*, 288, 1985.

[71] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society*, 39, 1977.

[72] F. Deprettere. *SVD and Signal Processing: Algorithms, Analysis and Applications*. Elsevier Science Publishers, Amsterdam, 1988.

[73] F. R. Deutsch. *Best Approximation in Inner Product Spaces*. Springer Verlag, first edition, 2001. ISBN 0387951563.

[74] E. Deza and M.-M. Deza. *Dictionary of Distances*. Elsevier, 2006.

[75] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of The 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2001)*, pages 269–274, 2001.

[76] I. S. Dhillon and J. Kogan, editors. *Workshop on Clustering High Dimensional Data and its Applications*, San Francisco, CA, May 2003. SIAM.

[77] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[78] I. S. Dhillon and S. Sra. Modeling data using directional distributions. Technical Report TR-03-06, Computer Sciences, The Univ. of Texas at Austin, January 2003.

[79] I. S. Dhillon and S. Sra. Generalized Nonnegative Matrix Approximations with Bregman Divergences. In *NIPS 18*, Vancouver, Canada, 2006.

[80] I. S. Dhillon and J. A. Tropp. Matrix Nearness Problems using Bregman Divergences, 2006. Submitted.

[81] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. Kumar R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[82] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, pages 131–138, 2002.

[83] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98, 2003.

[84] I. S. Dhillon, E. M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19(13):1612–1619, 2003.

[85] I. S. Dhillon, S. Sra, and J. A. Tropp. The Metric Nearness Problems with Applications. Technical Report TR-03-23, Computer Sciences, University of Texas at Austin, 2003.

[86] I. S. Dhillon, S. Sra, and J. A. Tropp. Triangle fixing algorithms for the metric nearness problem. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.

[87] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks*. Wiley, New York, 1996.

[88] A. J. Dobson. *An Introduction to Generalized Linear Models*. Chapman and Hall/CRC, second edition, November 2001.

[89] B. E. Dom. An information-theoretic external cluster-validity measure. Technical Report RJ 10219, IBM Research Report, 2001.

[90] D. Donoho and V. Stodden. When does nonnegative matrix factorization give a correct decomposition into parts? In *Neural Information Processing Systems*, 2003.

[91] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000.

[92] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[93] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 95:14863–14868, 1998.

[94] M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1–2), 1995.

[95] T. Feng, S. Z. Li, H-Y. Shum, and H. Zhang. Local nonnegative matrix factorization as a visual representation. In *Proceedings of the 2nd International Conference on Development and Learning*, pages 178–193, Cambridge, MA, June 2002.

[96] C. Fesel and A. Coutinho. Dynamics of serum IgM autoreactive repertoires following immunization: strain specificity, inheritance and association with autoimmune disease susceptibility. *Eur. Jour. Immunology*, 28:3616–3629, 1998.

[97] J. T. Foote and H. F. Silverman. A model distance measure for talker clustering and identification. In *Proceedings 1994 ICASSP*, volume S1, pages 317–320, 1994.

[98] D. Fradkin, I. B. Muchnik, and S. Streltsov. Image Compression in Real-Time Multiprocessor Systems using Divisive K-Means Clustering, 2003.

[99] A. G. Frenich, M. M. Galera, J. L. M. Vidal, D. L. Massart, J.R. Torres-Lapasió, K. De Braekeleer, J-H. Wang, and P. K. Hopke. Resolution of multicomponent peaks by orthogonal projection approach, positive matrix factorization and alternating least squares. *Analytica Chimica Acta*, 411:145–155, 2000.

[100] T. Fujiwara, S. Ishikawa, Y. Hoshida, K. Inamura, T. Isagawa, M. Shimane, H. Aburatani, Y. Ishikawa, and H. Nomura. Non-Negative Matrix Factorization of Lung Adenocarcinoma Expression Profiles. In *16th International Conference on Genome Informatics*, Yokohama Pacifico, Japan, December 2005.

[101] Y. Gao and G. Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21 (21):3970–3975, 2005.

[102] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, 2005.

[103] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the Natural Academy of Sciences*, pages 12079–12084, 2000.

[104] J. Ghosh. Scalable clustering methods for data mining. In Nong Ye, editor, *Handbook of Data Mining*, pages 247–277. Lawrence Erlbaum, 2003.

[105] A. Globerson and S. Roweis. Metric Learning by Collapsing Classes. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[106] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[107] Geoffrey J. Gordon. Generalized^2 linear^2 models. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 577–584, Cambridge, MA, 2003. MIT Press.

[108] J. C. Gower. Properties of Euclidean and non-Euclidean distance matrices. *Linear Algebra and its Applications*, 67:81–97, 1985. ISSN 0024-3795.

[109] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison Wesley, 1998.

[110] L. J. Gray and D. G. Wilson. Nonnegative factorization of positive semidefinite nonnegative matrices. *Linear Algebra and its Applications*, 31:119–127, 1980.

[111] L. Grippo and M. Sciandrone. On The Convergence of The Block Nonlinear Gauss-Seidel Method under Convex Constraints. *Operations Research Letters*, 26:127–136, 2000.

[112] D. Guillamet and J. Vitrià. Determining a suitable metric when using nonnegative matrix factorization. In *16th International Conference on Pattern Recognition*. IEEE Computer Society, 2002.

[113] D. Guillamet and J. Vitrià. Classifying faces with nonnegative matrix faces. In *CCIA*, Castelló de la Plana, Spain, 2002.

[114] D. Guillamet and J. Vitrià. Analyzing non-negative matrix factorization for image classification. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 116–119, 2002.

[115] D. Guillamet, M. Bressan, and J. Vitrià. A weighted nonnegative matrix factorization for local representations. In *CVPR*. IEEE, 2001.

[116] D. Guillamet, J. Vitrià, and B. Schiele. Introducing a weighted nonnegative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, October 2003. ISSN 0167-8655.

[117] A. Gunawardana and W. Byrne. Convergence Theorems for Generalized Alternating Minimization Procedures. *Journal of Machine Learning Research*, 6, 2005.

[118] J. Hannah and T. J. Laffey. Nonnegative factorization of completely positive matrices. *Linear Algebra and its Applications*, 55:1–9, 1983. ISSN 0024-3795.

[119] P. C. Hansen. The truncated SVD as a method for regularization. *BIT*, 27:534–553, 1987.

[120] R. A. Harshman and M. E. Lundy. The PARAFAC model for three-way factor analysis and multidimensional scaling. In Law et al., editor, *Research Methods for Multimode Data Analysis*, pages 122–215. Praeger, New York, 1984.

[121] J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[122] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, March 1972.

[123] Johan Håstad. Tensor rank is np-complete. *J. Algorithms*, 11(4):644–654, 1990.

[124] T. Hazan, S. Polak, and A. Shashua. Sparse image coding using non-negative tensor factorization. In *IEEE Conference on Computer Vision*, 2005.

[125] M. Heiler and C. Schnörr. Controlling sparseness in nonnegative tensor factorization. In *ECCV*, 2006.

[126] M. Heiler and C. Schnörr. Learning Sparse Representations by Non-Negative Matrix Factorization and Sequential Cone Programming. *JMLR*, 2006. To Appear.

[127] N. J. Higham. Matrix nearness problems and applications. In M. J. C. Gower and S. Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.

[128] G. W. Hill. Evaluation and inversion of the ratios of modified bessel functions. *ACM Transactions on Mathematical Software*, 7(2):199–208, June 1981.

[129] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer-Verlag, 2001.

[130] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. ACM SIGIR*. ACM Press, August 1999.

[131] P. O. Hoyer. Non-negative sparse coding. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[132] P. O. Hoyer. Modeling receptive fields with nonnegative sparse coding. *Neurocomputing*, 52–54:547–552, 2003.

[133] P. O. Hoyer. Nonnegative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[134] C. Hu, B. Zhang, S. Yan, Q. Yang, J. Yan, Z. Chen, and W.-Y. Ma. Mining Ratio Rules Via Principal Sparse Non-Negative Matrix Factorization. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 407–410, 2004.

[135] L. J. Hubert, P. Arabie, and J. Meulman. The representation of symmetric proximity data: Dimensions and classifications. *The Computer Journal*, 41(8), 1998.

[136] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102: 109–126, 2000.

[137] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley, 2001.

[138] A. Hyvärinen. Sparse Code Shrinkage: Denoising of Nongaussian Data by Maximum Likelihood Estimation. *Neural Computation*, 11(7):1739–1768, 1999.

[139] A. Hyvärinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[140] A. Hyvärinen, P. O. Hoyer, and E. Oja. Sparse Code Shrinkage: Denoising by Nonlinear Maximum Likelihood Estimation. In *NIPS*, pages 473–479. MIT Press, 1999.

[141] K. Inamura, T. Fujiwara, Y. Hoshida, T. Isagawa, M. H. Jones, C. Virtanen, M. Shimane, Y. Satoh, S. Okumura, K. Nakagawa, E. Tsuchiya, S. Ishikawa, H. Aburatani, H. Nomura, and Y. Ishikawa. Two subclasses of lung squamous cell carcinoma with different gene expression profiles and prognosis identified by hierarchical clustering and non-negative matrix factorization. *Oncogene*, 24:7105–7113, June 2005.

[142] P. Indyk. A Sublinear-time Approximation Scheme for Clustering in Metric Spaces. In *40th Symposium on Foundations of Computer Science*, 1999.

[143] P. Indyk. Sublinear time algorithms for metric space problems. In *31st Symposium on Theory of Computing*, pages 428–434, 1999.

[144] J-H. Oh J-H. Ahn, S. Kim and S. Choi. Multiple nonnegative-matrix factorization of dynamic pet images. In *ACCV*, 2004.

[145] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[146] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[147] M. W. Jeter and W. C. Pye. A note on nonnegative rank factorizations. *Linear Algebra and its Applications*, 38:171–173, 1981. ISSN 0024-3795.

[148] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[149] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings — good, bad and spectral. In *41st Annual IEEE Symp. Foundations of Computer Science*, pages 367–377, 2000.

[150] M. Kaykobad. On nonnegative factorization of matrices. *Linear Algebra and its Applications*, 96:27–33, 1987. ISSN 0024-3795.

[151] O. Kia. *Document Image Compression and Analysis*. PhD thesis, University of Maryland, Center for Automation Research, 1997.

[152] D. Kim, S. Sra, and I. S. Dhillon. A New Projected Quasi-Newton Approach for Nonnegative Least Squares Problem. Technical Report TR-06-54, Dept. of Comp. Sci., Univ. of Texas at Austin, 2006.

[153] D. Kim, S. Sra, and I. S. Dhillon. Fast Newton-type Methods for the Least Squares Nonnegative Matrix Approximation Problem. In *SIAM Data Mining*, 2007.

[154] P. M. Kim and B. Tidor. Subsystem Identification Through Dimensionality Reduction of Large-Scale Gene Expression Data. *Genome Research*, 13:1706–1718, 2003.

[155] Y. Kluger, R. Basri, J. Chang, and M. Gerstein. Spectral analysis of microarray cancer data. *Genome Research*, 13(4):703–716, 2003. To appear.

[156] D. E. Knuth. *The Art of Computer Programming*, volume 1: Fundamental Algorithms. Addison-Wesley, 3rd edition, January 1998.

[157] T. G. Kolda. Orthogonal tensor decompositions. *SIAM Journal of Matrix Analysis and Applications*, 23(1):243–255, 2001.

[158] T. G. Kolda and D. P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems*, 16(4):322–346, 1998. URL `citeseer.ist.psu.edu/article/kolda97semidiscrete.html`.

[159] T. Kosaka and S. Sagayama. Tree-structured speaker clustering for fast speaker adaptation. In *Proceedings 1994 ICASSP*, volume I, pages 245–248, April 1994.

[160] T. Kosaka and S. Sagayama. Tree-structured speaker clustering for speaker-independent continuous speech recognition. In *Proceedings 1994 ICSLP*, pages 1375–1378, September 1994.

[161] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978. Series: Quantitative Applications in the Social Sciences.

[162] B. Kulis, M. Sustik, and I. S. Dhillon. Learning low-rank kernel matrices. In *Proc. 23rd ICML*. ACM Press, 2006. To appear.

[163] W. Kun, Z. Nanning, and L. Weixiang. Natural image matting with nonnegative matrix factorization. In *IEEE International Conference on Image Processing*, volume 2, pages 1186–1189, September 2005.

[164] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions (with discussion). *J. of Comp. and Graphical Stat.*, 9(1):1–59, 2000.

[165] A. N. Langville and C. D. Meyer. Text mining using the nonnegative matrix factorization. SIAM Southeastern Section Annual Meeting, 2005. Talk.

[166] C. M. Lau and T. L. Markham. Factorization of Nonnegative Matrices–II. *Linear Algebra and its Applications*, 20:51–56, 1978.

[167] J. Laub and K-R. Müller. Feature discovery in Non-metric pairwise data. *Journal of Machine Learning Research*, 5:801–818, 2004.

[168] J. Lawrence, A. B-Artzi, C. DeCoro, W. Matusik, H. Pfister, R. Ramamoorthi, and S. Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. In *SIGGRAPH*, 2004.

[169] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient BRDF Importance Using a Factored Representation. In *SIGGRAPH*, 2004.

[170] C. Lawson and R Hanson. *Solving least squares problems*. Prentice-Hall, 1974.

[171] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice–Hall, Englewood Cliffs, NJ, 1974. Reissued with a survey on recent developments by SIAM, Philadelphia, 1995.

[172] D. D. Lee and H. S. Seung. Algorithms for nonnegative matrix factorization. In *NIPS*, pages 556–562, 2000.

[173] D. D. Lee and H. S. Seung. Unsupervised learning by convex and conic coding. In *NIPS*, pages 515–521. MIT Press, 1997.

[174] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, October 1999.

[175] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306(5701):1555–1558, 2004.

[176] J. S. Lee, D. D. Lee, S. Choi, and D. S. Lee. Application of nonnegative matrix factorization to dynamic positron emission tomography. In *3rd*

International Conference on Independent Component Analysis and Blind Signal Separation, San Diego, CA, December 2001.

[177] J. De Leeuw and G. Michailidis. Majorization methods in statistics. *Journal of Computational and Graphical Statistics*, 9:26–31, 2000.

[178] A. S. Lewis. Convex analysis on the Hermitian matrices. *SIAM J. Optim.*, 6(1):164–177, 1996. URL `citeseer.ist.psu.edu/lewis96convex.html`.

[179] Y. Li and A. Cichocki. Non-negative matrix factorization and its application in blind sparse source separation with less sensors than sources. In *Proceedings of XII International Symposium on Theoretical Electrical Engineering*, pages 285–288, Warsaw, Poland, 2003.

[180] Y. Li, A. Kummert, and A. Frommer. A linear programming based analysis of the CP-rank of completely positive matrices. *Int. J. Applied Math. Comput. Sci.*, 14(1):25–31, 2004.

[181] L.-H. Lim. Singular values and eigenvalues of tensors: a variational approach. In *Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP '05)*, pages 129–132, 2005.

[182] C.-J Lin. Projected-gradient methods for nonnegative matrix factorization, 2000.

[183] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, second edition, 1984.

[184] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for Biological Data Analysis: A survey. Technical report, INESC-ID, January 2004.

[185] O. L. Mangasarian. Normal solutions of linear programs. *Mathematical Programming Study*, 22:206–216, 1984.

[186] K. V. Mardia. *Statistical Distributions in Scientific Work*, volume 3, chapter Characteristics of directional distributions, pages 365–385. Reidel, Dordrecht, 1975.

[187] K. V. Mardia and P. Jupp. *Directional Statistics*. John Wiley and Sons Ltd., second edition, 2000.

[188] T. L. Markham. Factorizations of completely positive matrices. *Proceedings of the Cambridge Philosophical Society*, 69:53–58, 1971.

[189] T. L. Markham. Factorizations of nonnegative matrices. *Proceedings of the American Mathematical Society*, 32(1):45–47, March 1972.

[190] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent in Function Space. In *NIPS*, pages 512–518, 2000.

[191] P. McCullagh, , and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, second edition, 1989.

[192] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1997.

[193] G.J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley series in Probability and Mathematical Statistics: Applied Probability and Statistics Section. John Wiley & Sons, 2000.

[194] M. Meilă. Comparing clusterings by the variation of information. In *COLT*, 2003.

[195] R. R. Mettu and C. G. Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816–832, 2003.

[196] G. Meurant. *Computer Solution of Large Linear Systems*. North Holland, 1999.

[197] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The Discrete Basis Problem. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, LNCS, 2006.

[198] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. Müller. Fisher Discriminant Analysis with Kernels. In *Proceedings of IEEE Neural Networks for Signal Processing Workshop*, 1999. URL `citeseer.ist.psu.edu/mika99fisher.html`.

[199] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[200] V. Monga. Private Communication, February 2007.

[201] J. A. Mooney, P. J. Helms, and I. T. Jolliffe. Fitting mixtures of von Mises distributions: a case study involving sudden infant death syndrome. *Computational Statistics & Data Analysis*, 41:505–513, 2003.

[202] R. J. Muirhead. *Aspects of multivariate statistical theory*. John Wiley, 1982.

[203] K. E. Muller. Computing the confluent hypergeometric function, $m(a, b, x)$. *Numerische Mathematik*, 90(1):179–196, 2001.

[204] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[205] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A*, 135(3):370–384, 1972.

[206] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39 (2/3):103–134, 2000.

[207] F. J. Och. An efficient method for determining bilingual word classes. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 71–76, Bergen, Norway, 1999.

[208] M. F. Ochs, R. S. Stoyanova, F. Arias-Mendoza, and T. R. Brown. A new method for spectral decomposition using a bilinear bayesian approach. *Journal of Magnetic Resonance*, 137:161–176, 1999.

[209] E. Oja and M. Plumbley. Blind separation of positive sources using non-negative PCA. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, Nara, Japan, April 2003.

[210] P. Paatero. Least-squares formulation of robust nonnegative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

[211] P. Paatero. A weighted nonnegative least squares algorithm for three-way 'PARAFAC' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38:223–242, 1997.

[212] P. Paatero. The multilinear engine—a table-driven least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *Journal of Computational and Graphical Statistics*, 8(4):854–888, December 1999.

[213] P. Paatero and U. Tapper. Analysis of different modes of factor analysis as least squares fit problems. *Chemometrics and Intelligent Laboratory Systems*, 18(2):183–194, 1993.

[214] P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(111–126), 1994.

[215] P. Paatero, U. Tapper, P. Aalto, and M. Kulmala. Matrix factorization methods for analysing diffusion battery data. *Journal of Aerosol Science*, 22(Supplement 1):S273–S276, 1991.

[216] P. Paatero, P. K. Hopke, X-H. Song, and Z. Ramadan. Understanding and controlling rotations in factory analytic models. *Chemometrics and Intelligent Laboratory Systems*, 60:253–264, 2002.

[217] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 2000.

[218] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. Sys. and Sci.*, 43:425–440, 1991.

[219] L. Pardo. *Statistical Inference Based on Divergence Measures*. Chapman & Hall/CRC, 2006.

[220] M. Parnas and D. Ron. Testing metric properties. In *STOC*, pages 276–285, 2001.

[221] A. D. Pascual-Montano, P. Carmona-Saez, M. Chagoyen, and J.M. Carazo. Non-negative matrix factorization for gene expression and scientific texts analysis. In *ISMB*, 2003.

[222] P. Pauca, R. Plemmons, M. Giffin, and K. Hamada. Unmixing spectral data for space objects using independent component analysis and nonnegative matrix factorization. In *Proceedings Amos Technical Conference*, 2004.

[223] P. Pauca, F. Shahnaz, M. Berry, and R. Plemmons. Text mining using nonnegative matrix factorizations. In *SIAM Data Mining*, 2004.

[224] V. P. Pauca, J. Piper, and R. J. Plemmons. Nonnegative matrix factorization for spectral data analysis. Preprint, May 2005.

[225] P. S. Penev and J. J. Atick. Local feature analysis: a general statistical theory for object representation. *Computational Neural Systems*, 7:477–500, August 1996.

[226] J. H. Piater. *Visual Feature Learning*. PhD thesis, University of Massachussets, June 2001.

[227] J. Piper, V. P. Pauca, R. J. Plemmons, and M. Giffin. Object Characterization from Spectral Data using Nonnegative Factorization and Information Theory. Preprint, 2005.

[228] C. G. Plaxton. Personal Communication, 2003–2004.

[229] M. D. Plumbley. Conditions for nonnegative independent component analysis. *IEEE Signal Processing letters*, 9(6):177–180, June 2002.

[230] M. D. Plumbley. Algorithms for nonnegative independent component analysis. In *Unpublished*, 2002.

[231] Y. Qin, K. Oduyemi, and L. Y. Chan. Comparative testing of PMF and CFA models. *Chemometrics and Intelligent Laboratory Systems*, 61: 75–87, 2002.

[232] Z. Ramadan, B. Eickhout, X-H. Song, L. M. C. Buydents, and P. K. Hopke. Comparison of positive matrix factorization and multilinear engine for the source apportionment of particulate pollutants. *Chemometrics and Intelligent Laboratory Systems*, 66:15–28, 2003.

[233] C. R. Rao. *Linear Statistical Inference and its Applications*. Wiley, New York, 2nd edition, 1973.

[234] N. Rao, S. J. Shepherd, and D. Yao. Extracting characteristic patterns form genome-wide expression data by non-negative matrix factorization. In *IEEE Computational Systems Bioinformatics Conference*, 2004.

[235] E. Rasmussen. Clustering algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, New Jersey, 1992.

[236] J. A. Richards. *Remote Sensing Digital Image Analysis*. Springer-Verlag, New York, 1993.

[237] R. T. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1970.

[238] T. D. Romo, J. B. Clarage, D. C. Sorensen, and G. N. Phillips Jr. Automatic identification of discrete substates in proteins: singular value decomposition analysis of time-averaged crystallographic refinements. *Proteins*, 22:311–321, 1995.

[239] S. Rosset and E. Segal. Boosting density estimation. In *NIPS*, 2002.

[240] V. Roth, J. Laub, J. M. Buhmann, and K.-R. Müller. Going metric: Denoising pariwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, 2003.

[241] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of non-metric proximity data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(12), 2003.

[242] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290, December 2000.

[243] A. Rupar, A. Leonardis, and N. M. Kosta. Robust recognition using the tensor-rank principle. In *Vision with Non-Traditional Sensors*, 26th Workshop of the Austrian Association for Pattern Recognition, pages 45–52. Graz, September 2002.

[244] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, second edition, 2003.

[245] P. Sajda, S. Du, T. Brown, L. Parra, and R. Stoyanova. Recovery of Constituent Spectra in 3D Chemical Shift Imaging using Nonnegative Matrix Factorization. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 71–76, Nara, Japan, April 2003.

[246] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513–523, 1988.

[247] G. Salton and M. J. McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.

[248] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, 10, pages 285–295, 2001.

[249] E. Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Teil. Entwicklung willkürlichen Funktionen nach System vorgeschriebener. *Mathematische Annalen*, 63:433–476, 1907.

[250] I. J. Schoenberg. Remarks to Maurice Fréchet's article "Sur la definition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 36:724–732, 1935.

[251] M. Schutz and T. Joachims. Learning a Distance Metric from Relative Comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[252] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[253] P. Sebah and X. Gourdon. Convergence acceleration of series. Online, 2002. URL `http://numbers.computation.free.fr/Constants/constants.html`.

[254] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proc. of 8th Pacific Symposium on Biocomputing (PSB)*, 2003.

[255] F. Shahnaz, M. Berry, P. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. *J. on Information Processing and Management*, 42:373–386, 2006.

[256] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and Batch Learning of Pseudo-Metrics. In *Int. Conf. on Machine Learning (ICML)*, 2004.

[257] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 307–316. AAAI Press, 2000.

[258] A. Shashua and T. Hazan. Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision. In *ICML*, 2005.

[259] A. Shashua, R. Zass, and T. Hazan. Multiway Clustering using Super-symmetric Nonnegative Tensor Factorization. In A. Leonardis, H. Bischof, and A. Prinz, editors, *ECCV*, pages 595–608. Springer, 2006.

[260] K. Shimizu and K. Iida. Pearson type VII distributions on spheres. *Communications in Statistics: Theory & Methods*, 31(4):513–526, 2002.

[261] J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2001.

[262] P. Smaragdis and J. C. Brown. Nonnegative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, New Paltz, NY, October 2003.

[263] M. W. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.

[264] S. Sra. Efficient Large-Scale Linear Programming SVMs. Technical report, Computer Sciences, The Univ. of Texas at Austin, 2006.

[265] S. Sra and I. S. Dhillon. Multiplicative Update Algortihms for NNMA with Generalized Loss Functions. *Under Preparation*, March 2007.

[266] N. Srebro. *Learning with matrix factorizations*. PhD thesis, MIT, August 2004.

[267] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[268] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc 7th Natl Conf on Artificial Intelligence :*

Workshop of AI for Web Search (AAAI 2000), pages 58–64. AAAI, July 2000.

[269] A. C. Surendran and S. Sra. Incremental Aspect Models for Mining Document Streams. In *PKDD*, September 2006.

[270] B. Szatmáry, B. Póczos, J. Eggert, E. Körner, and A. Lőrincz. Nonnegative matrix factorization extended by sparse code shrinkage and weight sparsification algorithms. In *ECAI 2002, Proceedings of the 15th European Conference on Artificial Intelligence*, pages 503–507, Amsterdam, 2002. IOS Press.

[271] B. Szatmáry, G. Szirtes, A. Lőrincz, J. Eggert, and E. Körner. Robust hierarchical image representation using nonnegative matrix factorization with sparse code shrinkage preprocessing. *Pattern Analysis and Applications*, 2003. Accepted.

[272] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290:2319–2323, December 2000.

[273] L. B. Thomas. Rank factorizations of nonnegative matrices. *SIAM Review*, 16(4):393–394, 1974. Problem 73-14.

[274] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. Technical

report, Department of Health Research and Policy, Stanford University, 1999. URL `http://www-stat.stanford.edu/hastie/Papers/`.

[275] J. A. Tropp. *Topics in Sparse Approximation*. PhD thesis, The University of Texas at Austin, 2004.

[276] P. Tseng. An Analysis of the EM Algorithm and Entropy-Like Proximal Point Methods. *Mathematics of Operations Research*, 29:27–44, 2004.

[277] P. Tseng. Dual coordinate ascent methods for non-strictly convex minimization. *Mathematical Programming*, 59:231–247, 1993.

[278] K. Tsuda, S. Akaho, and K. Asai. The *em* Algorithm for Kernel Matrix Completion with Auxiliary Data. *JMLR*, 4(1), January 2004.

[279] M. A. O. Vasilescu and D. Terzopoulos. Multilinear image analysis for facial recognition. In *Int. Conf. Pattern Recognition (ICPR)*, Canada, August 2002.

[280] J. R. Wall. Rank factorizations of positive operators. *Linear and Multilinear Algebra*, 8:137–144, 1979.

[281] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. In D. P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*, LANL, pages 91–109. Kluwer, Norwell, MA, 2003.

[282] C. S. Wallace and D. L. Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, January 2000.

[283] M. Warmuth and K. Azoury. Relative Loss Bounds for On-line Density Estimation with the Exponential Family of Distributions. *Journal of Machine Learning Research*, 43(3):211–246, 2001.

[284] G. N. Watson. *A treatise on the theory of Bessel functions*. Cambridge Mathematical Library. Cambridge University Press, 2nd (1944) edition, 1996.

[285] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[286] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22:1255–1261, 2001.

[287] S. Wild, J. Curry, and A. Dougherty. Motivating nonnegative matrix factorizations. SIAM Linear Algebra Meeting, July 2003.

[288] A. T. A. Wood. Simulation of the von-Mises Distribution. *Communications of Statistics, Simulation and Computation*, 23:157–164, 1994.

[289] C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

[290] L. Xiao, J. Sun, and S. Boyd. A Duality View of Spectral Methods for Dimensionality Reduction. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 1041–1048, 2006.

[291] E. P. Xing and R. M. Karp. CLIFF: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics: Proceedings of 9th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 17:S306–S315, 2001.

[292] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, 2003.

[293] W. Xu, X. Liu, and Y. Gong. Document clustering based on nonnegative matrix factorization. In *SIGIR'03*, pages 267–273, Toronto, 2003. ACM.

[294] A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In *NIPS*, 2002.

[295] W. Zangwill. *Nonlinear programming: a Unified Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1969.

[296] R. Zdunek and A. Cichocki. Non-Negative Matrix Factorization with Quasi-Newton Optimization. In *Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC*, Zakopane, Poland, 2006.

[297] J. Zhang, L. Wei, Q. Miao, and Y. Wang. Image fusion based on non-negative matrix factorization. In *International Conference on Image Processing*, volume 2, pages 973–976, 2004.

[298] T. Zhang and G. H. Golub. Rank-one approximation to high order tensors. *SIAM Journal of Matrix Analysis and Applications*, 23(2):534–550, 2001.

[299] Ying Zhao and George Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, June 2004.

[300] S. Zhong and J. Ghosh. A comparative study of generative models for document clustering. In *Workshop on Clustering High Dimensional Data : Third SIAM Conference on Data Mining*, April 2003.

[301] S. Zhong and J. Ghosh. A Unified Framework for Model-based Clustering. *Journal of Machine Learning Research*, 4:1001–1037, November 2003.

[302] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. Technical report, Stanford University, 2004.

Vita

Suvrit Sra was born in Shimla, Himachal Pradesh, India on 29th October, 1976, the son of Ranjit Singh Sra and Ram Dulari Sra. The first 18 years of his life were spent in his beautiful hometown in the mountains. The next four years were devoted to earning a Bachelors of Engineering (Honors) degree in Computer Science from Birla Institute of Technology and Science (BITS) in Pilani. After a brief stint as a Software Engineer at Hughes Software Systems (now Flextronics), located in Gurgaon, India, he moved over to Austin, Texas to pursue a Ph.D. in Computer Science. He was a recipient of the MCD Fellowship from August 2000 to August 2004 at the University of Texas at Austin. After protracted indolence, he finally waded through the administrative morass and obtained his Masters in Computer Science degree in August 2006. After his doctorate, he plans to continue performing research in data mining and machine learning for the next few years.

Permanent address: 20 Powerhouse Road, Solan
H.P., India

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.